DOCUMENT RESUME

ED 057 812 LI 003 301

AUTHOR Silva, Georgette M.
TITLE Center for Information Services, Phase II: Detailed System Design and Programming, Part 7 - Text Processing, Phase IIA Final Report.
INSTITUTION California Univ., Los Angeles. Inst. of Library Research.
SPONS AGENCY National Science Foundation, Washington, D.C.
PUB DATE 1 Mar 71
NOTE 111p.; (1 Reference)

EDRS PRICE MF-$0.65 HC-$6.58
DESCRIPTORS *Automatic Indexing; *Computer Programs; Electronic Data Processing; *Information Centers; *Information Services; *Information Storage; Library Automation
IDENTIFIERS *Computer Software; University of California (Los Angeles)

ABSTRACT

Libraries, as well as larger information networks, are necessarily based upon the storage of information files consisting in many cases of written materials and texts such as books, serials, abstracts, manuscripts and archives. At the present stage of the "information explosion" no librarian can afford to ignore the contribution of modern computing equipment and advanced language processing methods to the problems involved in achieving higher levels of efficiency. This study is concerned with automatic text handling methods as required for an integrated system for mechanized information services in libraries. (Related documents are available as LI 003295 through LI 003300). (Author/NH)

ED057812

U.S. DEPARTMENT OF HEALTH,
EDUCATION & WELFARE
OFFICE OF EDUCATION
THIS DOCUMENT HAS BEEN REPRODUCED EXACTLY AS RECEIVED FROM THE PERSON OR ORGANIZATION ORIGINATING IT. POINTS OF VIEW OR OPINIONS STATED DO NOT NECESSARILY REPRESENT OFFICIAL OFFICE OF EDUCATION POSITION OR POLICY.

CENTER FOR INFORMATION SERVICES

PHASE II: DETAILED SYSTEM DESIGN AND PROGRAMMING

NSF GRANT GN-827

PHASE IIA FINAL REPORT

PART 7

TEXT PROCESSING

by

Georgette M. Silva

1 March 1971

Institute of Library Research
University of California
Los Angeles, California

LI 003 301

TABLE OF CONTENTS

		PAGE
I.	PRELIMINARY SPECIFICATIONS FOR THE CIS TEXT PROCESSING SYSTEM.	1
	Introduction.	1
	System Design	4
	System Functions.	6
	System Components	7
	Glossary.	19
	References.	20
II.	SOME TEXT PROCESSING PROCEDURES AND TECHNIQUES EXEMPLIFIED.	21
	Introduction.	21
	Program DEMO1	24
	Program TESTMT.	41
	Program INDX and Program CONCORD.	66

I.

PRELIMINARY SPECIFICATIONS FOR THE CIS TEXT PROCESSING SYSTEM

INTRODUCTION

This study is concerned with automatic text handling methods as required for an integrated system for mechanized information services in libraries. Libraries, as well as larger information networks, are necessarily based upon the storage of information files consisting in many cases of written materials and texts such as books, serials, abstracts, manuscripts and archives. At the present stage of the "information explosion" no librarian can afford to ignore the contribution of modern computing equipment and advanced language processing methods to the problems involved in achieving higher levels of efficiency.

Some of the implications of this new vision of library automation are contained in the following extract from the King report, Automation and the Library of Congress(1):

> "...the immediate objective of automation will be to solve the pressing problems that face research libraries, among which are the problems of bibliographic organization and control. In the long run, however, the most significant effect of automation will be the focussing of the services

of the library on the individual user for the optimal satisfaction of his research needs. Ideally an automated system should place the full resources of the library at the immediate disposal of the user. These resources should be more varied in nature and contain a richer set of statistical and bibliographical data than is practical to provide in conventional libraries."

Although there are a significant number of national programs concerned with the generating of cataloging and indexing data in machine-readable code, and a more restricted number of projects involved in creating information networks which would make the stored data readily available to users everywhere, not many libraries have progressed beyond the use of the computer for the production of catalogs, catalog cards, and for maintaining and storing serial records. There is a real need for the development and implementation of more sophisticated language processing methods before the "optimal satisfaction" of the user's research needs can be satisfied.

A great deal of material, from widely varying sources, is already available in a form machines can read. There are over 600 computerized typesetting installations in the U. S. and Canada alone, and very soon every printing house which wishes to remain competitive will have to produce a machine-readable version of a text as a natural by-product of the printing process. More and

more literary materials are being stored on magnetic tape, discs, or other media acceptable to computers, by individual scholars or institutions in America, Europe, India, Russia, Japan and the Near East. The demand for this type of data base is likely to increase, and research libraries will soon have to establish procedures to deal with them.

A text processing system, conceived as a subsystem of a mechanized library environment, must be capable of providing the scholar with automatic aids or automatic preliminaries to linguistic or literary analysis as well as incorporating appropriate techniques for other kinds of linguistic information processing. The latter includes such fields as content analysis, decision-making, information storage and retrieval, automatic abstracting and indexing.

Mechanical aids in humanistic research are word frequency counts, word indexes, concordances of varying kinds, and collating programs. General information retrieval problems in libraries require more emphasis upon techniques such as dictionary construction and dictionary look-up, statistical and syntactic language analysis methods, more sophisticated search strategies and matching procedures, and methods for semiautomatic on-line interactive processing.

The text processing system should be general enough to meet the needs of both the Humanities and the Sciences, and therefore have the capability to accept a wide variety of textual data bases in different languages, to carry out a wide variety of analyses in response to a user's request, and to have a flexible output

capability, i.e., to be able to satisfy a user's specifications regarding the format in which the retrieved information is to be presented. Furthermore, the system should be operational, economical and efficient. Since users' needs are never static, the design should allow for feed-back from the user, redefinition of objectives and modification. The design must also include the specification of criteria for evaluating the system, and plans for educating people to use the system.

SYSTEM DESIGN

General Requirements

The main objective of the system is to provide for storage, retrieval, retrospective searching, and reorganization of textual material already available in machine-readable form. However, since a great deal of natural language text of interest is normally only available in printed form, it is essential that the system include a facility for converting textual data of any kind into machine code.

Once the text is stored, say, on magnetic tape or discs, it is necessary to know the structure and organization of the file if any processing is to be done at all. The design specifications must therefore define the range of acceptable file structures, and provide facilities for identifying different areas in the file, as for example, title, author, journal, abstract, indexing terms. The programs must be so designed as to accept files stored on

diverse media with complex formats and widely varying character encoding schemes.

Any text manipulation invariably centers around some suitable combination of searching and matching. An efficient match of records with requests often requires some statistical measure of association. Language is rich in empirical statistics, and in processing a language to draw inferences, the statistician can provide advice and techniques for analyzing linguistic structures. The appropriate means must therefore be provided to accumulate statistics on the frequency of occurrence of predefined elements in a text, such as any symbol or group of symbols, words or string of words, word clusters, sentences, paragraphs, chapters, or any other structural unit.

Once the relevant machine function has been carried out, it should be possible to vary the output form and format of the retrieved information at will.

Further decisions are required regarding the suitability and relative efficiency of the various higher level programming languages presently available. There is no consensus of opinion amongst programmers as to which of the languages are to be preferred, and a series of controlled experiments designed to compare say, COBOL, PL/I and SNOBOL4, seems desirable.

The programs should be so designed as to make optimal use of machine speed and memory size. There should be no equipment limitations on the amount of data to be processed at any given time.

In order to attain a high degree of flexibility the system should be divided into autonomous routines, each corresponding to one of the main functions outlined above, and capable of being used singly or in combination with other subprograms to produce word frequency lists, word indexes, concordances, collated texts, or answers to any specialized request.

Furthermore, in order to make it easier for people not familiar with computer programming to make full use of the system, all available options should be controlled by parameters to be specified by the user and to be supplied to the program in the form of data.

SYSTEM FUNCTIONS

1. To retrieve information from a large variety of textual data bases available for support of research in the Humanities, Science and Engineering in response to specific requests.

2. To produce concordances and other word lists.

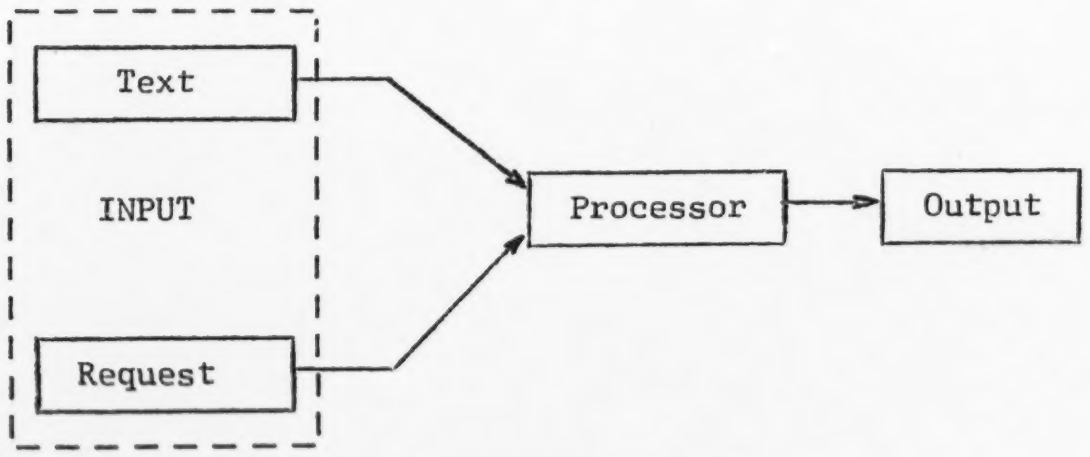
3. To accumulate statistics on frequency of occurrence of words and word strings.

4. To collate texts for the detection of differences and similarities.

5. To derive indexing terms based on a variety of clues, including frequency of occurrence, format, context, etc.

SYSTEM COMPONENTS

The following is a schematic representation of the components required in order to implement the functions outlined above:



The input consists of the textual data base and one or more requests.

The processor includes analysis, storage, retrieval and synthesis.

The output consists of word-lists, indexes, concordances, collated texts, or answers to specialized questions on demand.

1. Preparation of text for input

The first problem encountered in the field of language data processing is that of recording texts in a form suitable for machine manipulation. If the information is available in printed form, the obvious way to get it into the computer would be to have the computer read it. Unfortunately, multifont optical print scanners are not yet available, and, until they are, keypunching is the only

solution. Data of any kind can be keypunched either onto paper tape or onto standard 80-column cards. The choice of input medium is important since it is at the basis of subsequent decisions regarding character coding. A set of conventions for encoding symbols is required for any transformation of text to machine-readable code and for conversions from machine to machine when the two machines are not compatible. With most texts three distinct coding systems must be recognized: the punched card (or paper tape) code for source documents, the internal code of the computer used, and the definition of data in terms of it and the code of the output device. Most computer printers, for example, have character sets which do not necessarily coincide with that of the input device. The design of a capability for converting incoming text into a form acceptable to a computer therefore involves conceptual problems such as developing standards for symbol representation, magnetic tape or disc storage format, as well as the simpler problems relating to the physical nature of the input media used. Various aspects of the question have been treated by the following individuals and groups:

i. A Subcommittee of Committee Z39 of the USA Standards Institute (USASI) is concerned with development of standards for encoding bibliographic data in machine-readable forms.

ii. Another USASI subcommittee has developed the American Standard code for Information Interchange (ASCII) which

prescribes standards for bit configurations used to represent characters and functions.

iii. Members of the RAND Corporation staff, Kay & Ziehe (2,3) and Graves et al. (4), have taken a different appraoch to standards for bibliographic, typographic, and textual encoding. They propose "the catalog," which is a generalized data structure system designed to read texts with complex formats and to transform them into a standard magnetic tape storage format which can then be processed by standard programs or re-encoded according to standard conventions. The programs, although available on request, are written in the MAP assembly language for the IBM-7044, and are therefore only of limited use to other workers. In spite of the substantial effort which has already gone into this question, no universally acceptable proposal has as yet emerged. A great deal of work still remains to be done in order to decide on points like the following:

What should be standardized? Should the character encoding scheme be standardized, or rather the format for describing the tables which permit conversion from system to system? The same question applies to the magnetic tape or disc storage format. Should one adopt a standard archival format for textual data, or should

one rather seek to design generalized data management programs capable of dealing with any text no matter how complex its format?

iv. The solution proposed here is one which seems to me preferable in a library environment, namely, a program of the type suggested above, which would accept a simple formalized description of both the character encoding scheme and the file structure of the incoming text concurrently with the text at the time of processing and leave the input text unchanged rather than convert it into some standard format.

2. Basic programming procedures

2.1 Input package

Assuming that a document description and representation is available with each data base, the first routine required is an input package capable of accepting input on diverse media (cards, paper tape, magnetic tape, discs), and of decoding the logical records and identifying the various fields contained in the source document.

2.2 Text editing program

The next requirement for successful processing is a program to ensure that the characteristics set out in the description are consistently implemented.

2.3 Identification and referencing of information items

The basic procedure underlying much of the work in language processing consists of recognizing and tagging certain key elements, in most cases words or groups of words in the text, selected from a preconstructed list of acceptable items. Several conceptual problems need resolving before it is possible to arrive at an unambiguous interpretation of the information content:

How do we define a word, a phrase, a sentence?

How do we deal with abbreviations, proper nouns, compound forms of verbs, fixed phrases, hyphenated words?

How do we discriminate homographs?

How do we indicate semantic equivalences?

How do we recognize syntactic and semantic ambiguities?

A further major problem to be considered is the referencing system to be employed and the type of discriminants (tags) to be used. Any work requires some identification if later a particular phenomenon is to be traced to its place of occurrence in the file. The choice of the right discriminants is very important and should be guided by considerations of economy in relation to the kind of data to be retrieved.

Once the conceptual problems are resolved, the computer routine required here will simply break the incoming text into the requested units, decide on the basis of dictionary lookup whether the item is to be discarded or retained for further processing, append the appropriate numerical reference and context, if the item is to be concorded, and produce an output file suitable for sorting and subsequent processing.

A facility for counting the number of tokens in the text, for providing statistics on the number of sentences in the text, on wordlength and sentence length, should be built into this program.

2.4 Storage techniques

The predetermined lists from which the items to be processed are selected (see previous section) take the form of simple lists, dictionaries or thesauri, each of which need a different kind of storage structure to facilitate their efficient processing in high speed core.

Variable length word forms may either be stored in an array, or in some form of list structure. The latter may be a simple list, or a list of the pushdown storage type.

Dictionaries and thesauri are best stored in the form of lists of lists, or in a tree, which is a multilevel list structure.

Routines are therefore needed to provide the optimal storage method for the particular job.

2.5 Dictionary lookup

Dictionary lookup is an important and necessary step for most text analysis. The central operations in dictionary lookup are searching and matching. Search methods range from simple serial searches, which are fairly slow but can be speeded up by the use of directories, through "double sorting" (5), a well known scheme in computational linguistics, where the text is first indexed, and the index is subsequently matched against the dictionary entries, to the more elegant "tree" search strategies. Although in the latter case the storage structures involved are generally more complex, and the space required per unit of information larger, tree search strategies reduce the number of tests to be made when a word is to be looked up in the dictionary (and therefore the search time) by an important factor. While in sequential search one examines about half the entries of the dictionary, with binary search only n tests

are needed if the number of words in the dictionary is between 2^{n-1} and 2^n.

The most widely employed procedure for matching elements of the text against dictionary entries is to cut the text at blanks (and possibly punctuation marks), and search the dictionary for the letter strings thus obtained. This method does not allow for the recognition of discontinuous stretches of text which, from the point of view of the lexicographer, must be retained as a single unit. Examples of such strings are: 'on the go,' 'post box,' 'in spite of,' 'in view of,' M.D., and innumerable others.

Several matching techniques are available to cope with this problem. One such method, "the longest from the front" (5), matches the dictionary entries against the text, which is now regarded as a string of characters made up of letters, punctuation marks, special symbols, and blanks.

SMART (6), a carefully constructed retrieval system operating in a laboratory environment, makes use of even more sophisticated matching techniques, some employing methods for phrase and synonym recognition, others based upon weighted word-stem matching and statistical concept association measures, and still

others based upon hierarchical expansions of relations between concepts.

Work on problems related to searching techniques and matching procedures is by no means exhausted, and there is need for further research in design and evaluation in this area. It is therefore not possible, at this stage, to indicate precisely what specific routines should be included in the final design of the proposed text processing system.

2.6 Sorting program

Sorting a number of items into alphabetic or numerical order is one of the commonest and best understood but by no means straight-forward operations in automated language processing. If the number of items to be sorted is small, they can be placed in storage all at once and reordered by means of a "bubble sort" or some tree sorting method. If a file is too large to be stored internally, it is far more convenient to use the general purpose sort program usually supplied by the manufacturer.

Both methods for internal and external sorting should be made available to the user of the system.

Some of the problems involved in sorting alphabetic information are related to diacritics and to spelling conventions.

Sorting programs using standard collating sequences will not arrange words into alphabetical order unless all special, non-alphabetic characters such as apostrophes, $-signs, and diacritical marks are first removed from the words to be sorted. Consider the following names which could appear in the order shown in a card catalog: Oborn, O'Boyle, Obradovic, O'Bree. As far as the computer is concerned Ob and O'B are <u>non-identical</u> strings of characters and so are Obr and O'Br. In order to make them identical, they must be changed in some suitable way <u>before</u> the sort process. The original form of the words will be reestablished after the sort process.

Since this is a problem which needs to be dealt with before sorting takes place, a suitable routine must be incorporated in the program discussed in the section on Identification and Referencing of Information Items.

A similar, but much more complex problem is that of sorting surnames with prefixes such as A', Ap, Fitz, M', Mac, Mc, Saint, St. San, and many others. The

confusion which may arise between Roman numerals and capital letters is another problem that needs to be considered.

Records prepared for sorting invariably consist of a number of alphabetic and numeric fields. A means must be provided to specify the desired sequence of the sort fields of output records.

From the above discussion it is clear that more flexibility is required than most general purpose sorting procedures provide.

2.7 Text collating program

The aim of this procedure is to bring lineation of all texts to be compared into conformity with that of one text designated "copy text" and to print out all variants in the compared texts beneath the copy text in the form of a Manly and Rickert card.(7)

2.8 Output routine

This component of the system should have the capability of presenting the results of the particular procedure employed in the form and format specified by the user. This will generally consist in listing the results of a search or a sort in some suitable format.

The main function of this program will be to interpret the output string of symbols and to reconstruct from them everything of significance about the text they represent. For example, there may be need to restore special characters to words, if previously removed to facilitate sorting, and to translate all special coding into normal typographical conventions in such a way that the output becomes acceptable to publishers; on the other hand, the program may have to insert suitable flags, shifts, and other marks in order to prepare the output for automatic typesetting.

GLOSSARY

A concordance is an alphabetically ordered list of the occurrences of either all, or a selection of items in a text, each occurrence surrounded by an appropriate portion of its context together with a numerical reference to each passage where the item occurs.

A dictionary is a file of information about small, repetitive elements of a language. The information is arranged in such a manner as to make it easy to find the properties of an element, given its shape.

A word frequency list is simply the list of all items in a text arranged in decreasing order of frequency.

A word index is an alphabetically ordered list of all items in a text with complete numerical reference to each passage where the term occurs, plus a frequency count for each item.

REFERENCES

1. King, Gilbert W., et.al., Automation and the Library of Congress (Washington, Library of Congress, 1963), pp. 2-3.

2. Kay, M. and T. Ziehe, Natural Language in Computer Form. RAND Corp., Santa Monica, California, February 1965. (Report No. RM-4390-PR) (AD-456948).

3. Kay, M. and T. Ziehe, The Catalog: A Flexible Data Structure for Magnetic Tape. RAND Corp., Santa Monica, California, October 1965. (Report No. RM-4645-PR).

4. Graves, Patricia A., David G. Hays, Martin Kay, and Theodore W. Ziehe, Computer Routines to Read Natural Text with Complex Formats. RAND Corp., Santa Monica, California, August 1966, 141 p. (Report No. RM-4920-PR) (AD-637 303).

5. Hays, D. G., Introduction to Computational Linguistics, Elsevier, 1967.

6. Salton, G., Keen, E. M., Lesk, M., "Design Experiments in Automatic Information Retrieval". In: Kochen, M., ed. The Growth of Knowledge: Readings on Organization and Retrieval of Information. Wiley, New York, 1967.

7. Manly, John M. and Edith Rickert, The Text of Chaucer's Canterbury Tales, Chicago, 1940, II, 3-12.

II.

SOME TEXT PROCESSING PROCEDURES AND TECHNIQUES EXEMPLIFIED

I. INTRODUCTION

The work described in this report was carried out within the framework of the CIS Development Program, and is relevant to the Text processing module. A general description of text handling methods suitable for an integrated system for mechanized information services in libraries was given in an earlier report.[1]

The present report contains four working programs each illustrating some of the basic procedures and techniques most commonly needed in automatic language processing. The four programs are listed below in order of complexity of the tasks performed:

Program DEMO1
Program TESTMT
Program INDX
Program CONCORD

Each program is capable of carrying out one or more of the tasks listed below:

1. To accept text either from magnetic tape or from cards.
2. To identify textual units such as single characters, letters, numbers, punctuation marks, single words, multi-word phrases, and major divisions within text, such as volumes, chapters, paragraphs, sentences and word order within sentence or, in the case of bibliographic records, title, author, abstract, and many others.
3. To carry out various character conversions, such as converting from one encoding system to another, or, simply giving binary, hexadecimal and decimal equivalents.

4. To reject words on an exclusion list, if so desired, to sort words by frequency, and retrieve words in context.

The following is a brief summary of the purposes and capabilities of each program:

Program DEMO1 reads text from magnetic tape, breaks it into individual words, sorts them alphabetically and in order of decreasing frequency, and lastly, prints each word surrounded by context in order of occurrence in the text. All these tasks are carried out in internal memory. The program performs these tasks on a fairly elementary methodological level. The sort, for example, is simply what is termed a "bubble sort". The text must be limited to 10,000 characters. The latter condition is not serious, however, since this is more than adequate to cope with short texts such as abstracts. The main virtue of this program is that it consists of one deck, and carries out all the tasks in one run. It may, therefore, be useful as a demonstration program for beginners.

Program TESTMT is somewhat more sophisticated and has a few additional capabilities: it does character conversion by table lookup, retrieves index terms in context, and uses a binary tree sort for the dictionary and frequency sort. These operations are carried out in internal memory, and the amount of text processed at any given time is still limited.

Program INDX is an adaptation of Sally Yeates Sedelow's program INDEX fully described in Automated Language Analysis

1967 - 1968[1]. The program reads text from cards, and produces a word index and a word frequency list. There are several reasons for which I chose to implement Mrs. Sedelow's program, rather than another. Firstly, it was designed with a large number of applications in veiw. Secondly, it was available, well documented, and claimed to be operational. Lastly, there are no limitations on the volume of material it can cope with at any time.

Program CONCORD makes a concordance in KWIC format. It was designed to eventually accept text from cards in the same format as program INDX. Both program INDX and program CONCORD use the IBM/360 utility sort package. Linda Clark from CCN has been responsible for the specifications and the implementation of the appropriate options of each case.

In most cases only macro-flowcharts have been included. The listings supplied with each program contain numerous comments and also represent the entire deck setup, JCL included. They are meant to temporarily take the place of both micro-flowcharts and descriptions of deck setup.

Considerations of efficiency have at all times been present in my mind while writing these programs, but have never been my primary concern. My main goal was to provide a set of working programs illustrating some of the fundamental, even if fairly rudimentary tasks and techniques in automatic language processing.

---

[1]Sedelow, Sally Y. (Principal Investigator), et. al. Automated Language Analysis 1967 - 1968, report on Research for the Period March 1, 1967 - February 29, 1968. Chapel Hill: University of North Carolina, Department of Info rmation Science.

II. PROGRAM DEMO1

1. Identification

Title	DEMO1
User Group	Institute of Library Research
Category	CIS Text Processing
Source Language	PL/1
Machine Configuration	IBM/360 Model 91, a tape, printer
Space Requirements	Step Region:
	PL1L 162K
	GO 150K
	Core Used 144K
Authors	G. Silva and H. Borko
Date	March 1970

2. Purpose

This program illustrates some of the most basic procedures needed in the processing of natural language data, namely, how to accept records from magnetic tape, how to identify single words, how to use an exclusion list, how to sort words into alphabetical and frequency order, and finally, how to prepare a keyword-in-context-type record.

Since the program was designed as a teaching tool at a very elementary level, the programming techniques employed are suitably uncomplicated. We have refrained from using preprocessor and pointer variables, and have settled on a "bubble sort". We have, however,

employed a "binary search" technique for matching words against the exclusion list, and not the simpler, but much slower sequential search. All operations are carried out in core, and the amount of text that can be processed at any one time is therefore limited.

The Aspen tape, containing documents from the California Constitution was used for testing purposes.

3. Input Specifications

Input is accepted from the Aspen tape which has the following specifications:

1. 9 track tape
2. 800 BPI
3. No header or trailer labels
4. First file on tape ILR080
5. Volume serial number is ILR080
6. Data Set name is CONCAL
7. Record format is fixed blocked (FB)
8. Logical record length is 85 bytes
9. Block size is 1615

The DD card used to read the tape is:

```
//GO.ASPEN  DD DSNAME=CONCAL,UNIT=TAPE9,VOLUME=SER=ILR080,
//  DCB=(RECFM=FB,BLKSIZE=1615,LRECL=85),DISP=OLD,LABEL=(1,NL,,IN)
```

4. File Description

The file contains fixed blocked records throughout with the following record format:

Positions 1 - 5, ascending document numbers,

Positions 6 - 85, text.

The first record of each document contains indicative data for that document instead of text.

5. Checkout

5.1 Timing

The program processed 6 documents from the Aspen tape, (containing a total of 429 words before exclusion, and 240 words after exclusion) with the following results:

Step Name	Step CPU Seconds
PL1L	3.49s
GO	4.19s
Load Timing	.41s
Total processing time	3.78s

Average processing time=8.8 ms/input word or

.88 s/100 input words

5.2 Error Checking

None

5.3 Cost

The cost of running this program was:

PL1L	$2.29
GO	$2.61
Total	$4.90

5.4 Remarks

The amount of text that can be processed in core is limited by the size of arrays 'VOCAB' and 'IT'. The maximum size allowable by PL1 would be approximately 1600x20 bytes. Assuming the average word length to be 5 characters, and allowing for one blank between

words, the maximum permissible length for one section is 9600 bytes.

6. Program Description

Main steps

1. HEADPRO: Read and print one document with heading and sub-heading centered and underlined, text formatted as found on magnetic tape but without document number appearing at beginning of each line.
2. WRDLST: Call subroutine NI to identify words. Store these words as character strings of length 20 in variable length array VOCAB.
3. Store array VOCAB on magnetic tape in file LEX, declared a print file with following specifications:

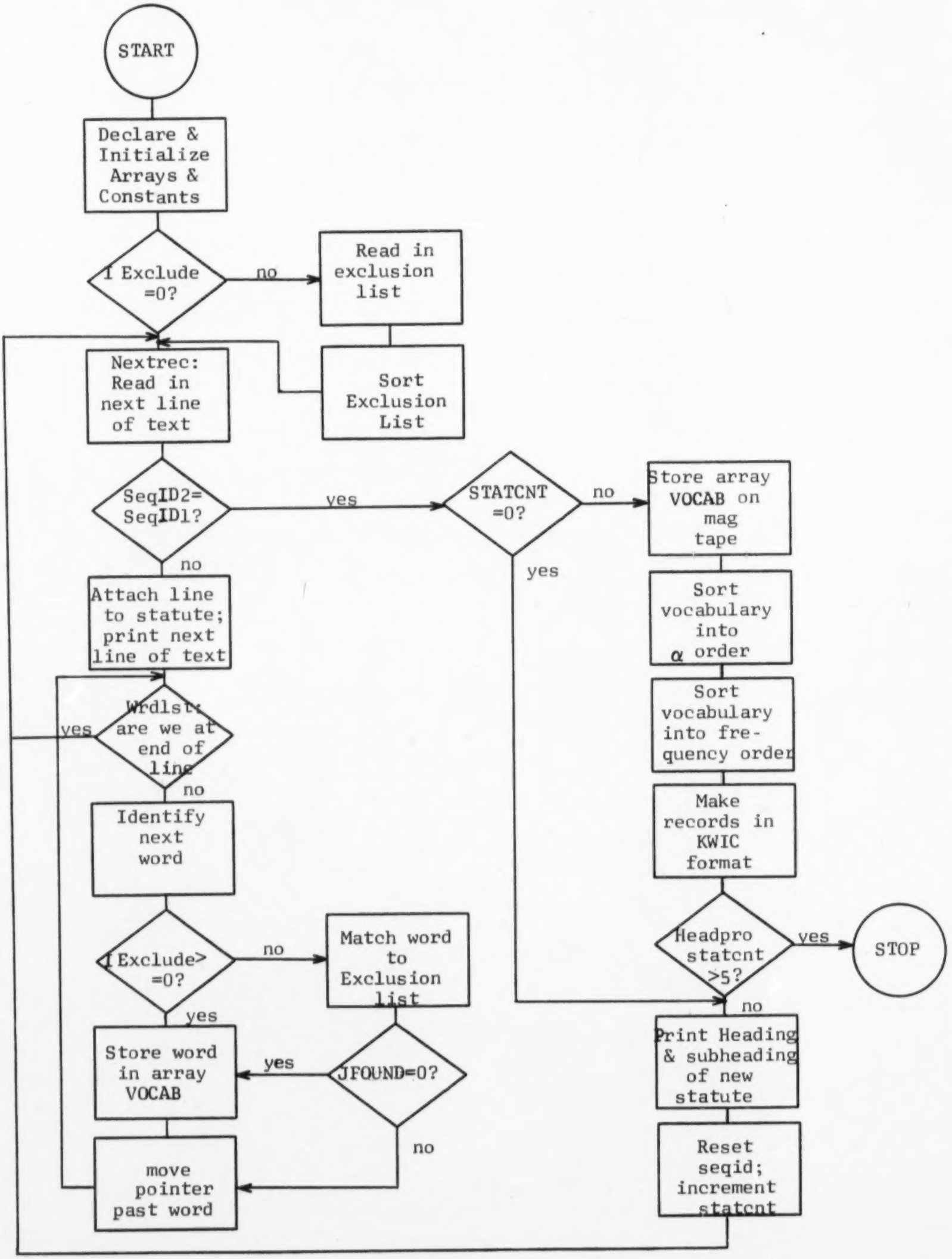
Logical record length	20
Block size	4000
Record format	VBA

4. Call SORT, subroutine to sort words into alphabetical order (ascending order). Precede call command by statement ORDER=1.
5. Call FREQCNT subroutine to do the following:
   a. count frequency of each word
   b. store word with frequency count preceeding it as character string of length 20 in variable-length array 'IT'
   c. print list of words in alphabetical order with frequency count for each word.
6. Call SORT subroutine to sort words in order of descending frequency. Precede call command by the statement ORDER=2.

7. Print list of words in order of descending frequency under heading 'FREQUENCY LIST'.
8. Call CONTEXT subroutine to process the text of the document now stored in STATUTE:
   a. call NI subroutine to identify the next word
   b. find length of word
   c. print word, underlined in KWIC format (key word in context) so that word is centered in line with 49 characters of context proceeding it and 49 characters following it.
9. An exclusion condition is written into the program which caused the following modifications when IEXCLUDE is set not equal to zero:
   a. before the first record is read in, the READIN subroutine is called to read into memory a list of terms, typed one per card in a data deck whose last card has a zero in column one
   b. call sort subroutine to sort exclusion list into alphabetical order
   c. print this under heading 'Exclusion list'
   d. after a word has been identified and before it is sorted (see #2) call subroutine BIS (a binary search) to match word against exclusion list
   e. only those words not found in exclusion list are added to array VOCAB, so that remaining operations are performed only on words not found in exclusion list.

Note that this program is not up to a normal standard for several reasons. One of them has to do with subroutine SORT which arranges records in either ascending or descending order, according to whether ORDER=1 or ORDER=2. This is fine for sorting words into dictionary order. However, when this sort is used to arrange words by frequency, not only the frequency count, but also the words, come out in descending order. The accepted method of presentation for a frequency list is to have words within frequency arranged in dictionary order. In order to achieve this, the present routine would have to be modified. Another defect is to be found in subroutine CONTEXT, which should, but does not, produce output records on constant length (120 characters). This is due to a minor programming error, which, had time permitted, could have been rectified.

7. Flow Chart


START
Declare & Initialize Arrays & Constants
I Exclude =0?
no
Read in exclusion list
Nextrec: Read in next line of text
Sort Exclusion List
SeqID2= SeqID1?
yes
STATCNT =0?
no
Store array VOCAB on mag tape
yes
Sort vocabulary into α order
Sort vocabulary into frequency order
Make records in KWIC format
Headpro statcnt >5?
yes
STOP
no
Print Heading & subheading of new statute
Reset seqid; increment statcnt
no
Attach line to statute; print next line of text
Wrdlst: are we at end of line
yes
no
Identify next word
I Exclude> =0?
no
Match word to Exclusion list
yes
JFOUND=0?
yes
Store word in array VOCAB
no
move pointer past word

8. Program Listing

```
//NRB4C9HB   JOB   '15,2500,R',BORKO
// EXEC PL1
//PL1L.SYSIN DD *
  (SUBSCRIPTRANGE):(STRINGRANGE):
  DEMO1:  PROC OPTIONS(MAIN);
          DCL (DUMMY) CHAR(85)VAR INIT ((85)' '),
      (SEQID1, SEQID2) CHAR(5) INIT ('00000'),
          ULINE CHAR(80)VAR INIT((80)'_'),
      WRDLGTH BIN FIXED,
      STATCNT BIN FIXED,
      WRDCNT BIN FIXED,
          VOCAB(500) CHAR (20)VAR;
          DCL ORDER BIN FIXED;
          DCL LEN BIN FIXED;
          DCL(II, NW)BIN FIXED;
          DCL (FRSTLTR, LSTLTR) BIN FIXED;
          DCL LEX PRINT;
                  DCL K BIN FIXED;
                  DCL STATUTE CHAR (10000)VAR;
  DCL TEMP CHAR (20)VAR;
  DCL EXCL_LIST (100) CHAR (82) VAR;
          DCL IT(500) CHAR(30)VAR;
  ON ENDFILE (ASPEN) GO TO ENDCON;
  STATCNT = 0;
          NW = 0;
          STATUTE = '';
PUT PAGE;
          GET LIST(IEXCLUDE);   GET SKIP;
IF IEXCLUDE=0 THEN GO TO NEXTREC;
CALL READIN(EXCL_LIST, NO_EXCL);
          ORDER =1;
CALL SORT (NO_EXCL, EXCL_LIST, ORDER);
          PUT SKIP LIST('EXCLUSION LIST');
          DO J = 1 TO NO_EXCL;
          PUT SKIP LIST(EXCL_LIST(J));
          END;
NEXTREC:  /*READ IN NEXT LINE OF TEXT*/
          DUMMY = (85)' ';
          READ FILE(ASPEN) INTO (DUMMY);
          SEQID2 = SUBSTR(DUMMY, 1, 5);
          DUMMY = SUBSTR(DUMMY, 6);
          LEN = 81;
          LOOP1:  LEN = LEN-1;
                  IF SUBSTR (DUMMY,LEN,1) = ' ' THEN GO TO LOOP1;
/* LEN EQUALS LENGTH OF LINE */
          IF SEQID2>SEQID1 THEN DO;
          IF STATCNT>0 THEN GO TO LSTPRNT;
          GO TO HEADPRO;
          END;
          STATUTE = STATUTE||SUBSTR(DUMMY, 1, LEN+1);
          PUT SKIP LIST(DUMMY);  /* PRINTS NEXT LINE OF TEXT  */
          GO TO WRDLST;
HEADPRO:  /*PROCESSES HEADING AND SUBHEADING OF STATUTE*/
          SEQID1 = SEQID2;
          IF STATCNT>2 THEN GO TO ENDCON;
 /* PROCESSES ONLY ONE STATUTE AT A TIME*/
          STATCNT = STATCNT+1;
          NW = 0;
          STATUTE = '';
          PUT EDIT (DUMMY) (SKIP(3), COL(30), A);
                  PUT EDIT (ULINE) (SKIP(0), COL(30), A(LEN));
```

Program Listing (Continued)

```
        READ FILE(ASPEN) INTO (DUMMY);
        DUMMY = SUBSTR(DUMMY, 6);
        PUT EDIT (DUMMY) (SKIP(2), COL(25), A);
        LEN = 81;
        LOOP2:  LEN = LEN-1;
                IF SUBSTR(DUMMY, LEN, 1)= ' ' THEN GO TO LOOP2;
                PUT EDIT (ULINE) (SKIP(0), COL(25), A(LEN));
               PUT SKIP;
        GO TO NEXTREC;
WRDLST:  /*STORE WORDS IN ARRAY VOCAB*/
        K = 1;
        NOW = 20;
        DO J=1 BY 1 TO NOW WHILE (K<=LEN);
        CALL NI(TEMP, FRSTLTR, LSTLTR, DUMMY, LEN, K);
IF IEXCLUDE = 0 THEN GO TO ADD;
      CALL BIS(EXCL_LIST, NO_EXCL, TEMP, JFOUND);
      IF JFOUND = 1 THEN GO TO OUT;
ADD:  NW = NW + 1;
    VOCAB(NW) = TEMP;
OUT:  K = LSTLTR + 1;
        END;
       GO TO NEXTREC;
LSTPRNT:
        DO K=1 TO NW;
       PUT FILE (LEX) LIST (VOCAB(K));
        END;
/*  PUT ORDER = 1 TO SORT IN ASCENDING ORDER */
        ORDER = 1;
        CALL SORT(NW, VOCAB, ORDER);
        PUT SKIP LIST('SORTED LIST');
        CALL FREQCNT(NW, VOCAB, II, IT);
/*  PUT ORDER = 2 TO SORT IN DESCENDING ORDER */
        ORDER = 2;
        CALL SORT(II, IT, ORDER);
        PUT SKIP LIST('FREQUENCY LIST');
        DO J = 1 TO II;
        PUT SKIP LIST(SUBSTR(IT(J), 1, 9), SUBSTR(IT(J), 10));
        END;
CALL CONTEXT(STATUTE);
              GO TO HEADPRO;
 BIS:
        PROC( LIST, LL,   ITEM, JFOUND);
 /*  BIS STANDS FOR " BINARY SEARCH "   */
        /*  BIS RETURNS JFOUND=1 IF ELEMENT FOUND, ZERO, IF NOT  */
        DCL ITEM CHAR(*)VAR;
        DCL LIST(*        ) CHARACTER(* )VAR;
        LOWER=1;
        IUPPER= LL+1;
 B10:
        MID = (LOWER + IUPPER)/2;
        IF ITEM = LIST(MID) THEN DO;
        JFOUND=1;
        RETURN;
        END;
        IF ITEM > LIST(MID) THEN DO;
            IF(MID = LOWER)|(MID = IUPPER ) THEN DO;
            JFOUND=0;
            RETURN;
            END;
        LOWER = MID;
        GO TO B10;
```

Program Listing (Continued)

```
                                        END;
            IF (MID = LOWER)| ( MID = IUPPER) THEN DO;
            JFOUND=0;
            RETURN;
            END;
          IUPPER=MID;
          GO TO B10;
 END BIS;
CONTEXT:  PROC (INPUT);
/*  PRINT WORDS OF TEXT IN CONTEXT  */
         DCL WORD CHAR(20)VAR INIT((20)' '),
         INPUT CHAR (*) VAR,
         TEXT CHAR(120)VAR,
         ULINE CHAR(80)VAR INIT((80)'_'),
(IPLGTH, FRSTLTR, LSTLTR, J , K, LENG, IBLNK) BIN FIXED,
         (FRONT, BACK, IFRONT) CHAR(50) VAR INIT((50)' ');
         IPLGTH = LENGTH(INPUT);
STEP1:
/*  FIND NEXT WORD AND WORDLENGTH  */
         K = 1;
SEARCH:
         FRONT=(50) ' ';
         BACK = (50) ' ';
         IFRONT = (50) ' ';
         DO J=1 BY 1 WHILE(K<=IPLGTH);
CALL NI(WORD, FRSTLTR, LSTLTR, INPUT, IPLGTH, K);
         LENG   = LSTLTR - FRSTLTR;
STEP2:
/*  CONTEXT BEFORE WORD  */
         IF FRSTLTR<51 THEN DO;
         IFRONT = SUBSTR(INPUT, 1, FRSTLTR-1);
         IBLNK = 51-FRSTLTR;
         FRONT=SUBSTR(FRONT,1,IBLNK)||IFRONT;
         GO TO STEP3;
         END;
         FRONT = SUBSTR(INPUT, FRSTLTR-50, 50);
STEP3:
/*  CONTEXT AFTER WORD  */
         IF IPLGTH<FRSTLTR+LENG+50 THEN DO;
         BACK = SUBSTR(INPUT, FRSTLTR+LENG);
         GO TO STEP4;
         END;
         BACK = SUBSTR(INPUT, FRSTLTR+LENG, 50);
STEP4:
/*  PRINT WORD IN CONTEXT  */
         TEXT = FRONT || WORD || BACK;
         PUT SKIP LIST(TEXT);
         PUT EDIT (ULINE)(SKIP(0), COL(51), A(LENG));
         K = LSTLTR+1;
         GO TO SEARCH;
         END;
RETURN;
END CONTEXT;
FREQCNT:
         PROC(NW, VOCAB, II, IT);
         DCL (NW, II)BIN FIXED,
         IT(*)CHAR(*)VAR,
         VOCAB(*)CHAR(*)VAR,
         (IFREQ, I)BIN FIXED;
        DCL (TEMP1, TEMP2) CHAR(20) VAR;
        IFREQ=1;
```

Program Listing (Continued)

```
        I=1;
         II=0;
        TEMP1 = VOCAB(I);
FREQLOOP: TEMP2 = VOCAB(I+1);
        IF TEMP1 ¬= TEMP2 THEN GO TO LIST;
        IFREQ = IFREQ + 1;
        I = I+1;
         IF I>NW THEN GO TO LIST;
        GO TO FREQLOOP;
LIST:
         II=II+1;
         IT(II) = IFREQ||TEMP1;
         PUT SKIP LIST(IFREQ, TEMP1);
        TEMP1 = TEMP2;
        IFREQ = 1;
        I = I+1;
         IF I>NW THEN GO TO FREQSORT;
        GO TO FREQLOOP;
FREQSORT:
         RETURN;
         END FREQCNT;
NI:
         PROC(TEMP, FRSTLTR, LSTLTR, DUMMY, LEN, K);
         DCL TEMP CHAR(*)VAR,
         (FRSTLTR, LSTLTR, LEN)BIN FIXED,
         DUMMY CHAR(*)VAR;
/* LOOPS WHEN PUNCTUATION ENCOUNTERED */
         DO FRSTLTR=K TO LEN WHILE
      (INDEX(' ,.;?"=()-', SUBSTR(DUMMY, FRSTLTR, 1)) ¬=0);
         END;
/* LOOPS WHEN NO PUNCTUATION ENCOUNTERED, TO FIND WRDLGTH */
       DO LSTLTR = FRSTLTR+1 TO LEN WHILE
      (INDEX(' ,.;?"=()-', SUBSTR(DUMMY, LSTLTR, 1)) = 0);
         END;
         TEMP = SUBSTR(DUMMY, FRSTLTR, (LSTLTR-FRSTLTR));
         RETURN;
         END NI;
READIN:
         PROC(LINE,NUM);
/*  READS IN A LIST OF TERMS TYPED ONE PER CARD  */
/* DATA DECK MUST END WITH A ZERO IN COL 1  */
/* NO OTHER FORMAT RESTRICTIONS ARE IMPOSED  */
         DCL LINE (*)CHAR(*)VAR;
         DCL I FIXED BIN;
         N=0;
R1C:
         N=N+1;
         GET EDIT(LINE(N)) (A(80));
         IF LINE(N) = '0' THEN GO TO R50;
/* NORMALIZE CARD FORMAT  */
/* ADD BLANK AT THE BEGINNING AND AT THE END OF  EACH LINE  */
         LINE(N) = ' '||LINE(N)||' ';
/* REDUCE MULTIPLE BLANKS TO ONE BLANK  */
R2C:
         I = INDEX(LINE(N),'  ');
         IF I¬=0 THEN DO ;
              IF I=LENGTH(LINE(N))-1  THEN DO ;
              LINE(N) = SUBSTR(LINE(N),1,I);
              GO TO R30;
              END;
         LINE(N) = SUBSTR(LINE(N),1,I)||SUBSTR(LINE(N),I+2);
```

Program Listing (Continued)

```
        GO TC R20;
        END;
 R30:
/* REMOVE LEADING ANC TRAILING BLANKS */
        LINE(N)=SUBSTR(LINE(N), 2, LENGTH(LINE(N))-2);
        GO TO R1C;
 R50:
        NUM = N-1;
        RETURN;
 END READIN;
 SORT:
        PROC(N, ITEM, ORDER);
        DCL ORDER BIN FIXED;
        DCL ITEM(* )CHAR(* )VAR;
        DCL TEMP CHAR(80)VAR;
        NUM = N-1;
        J=0;
 S1:    J=J+1;
        IF J>NUM THEN GO TO S10;
        IF ORDER = 2 THEN DO;
        IF ITEM(J) < ITEM(J+1) THEN GO TO S2;
        GO TO S1;
        END;
        IF ITEM(J) < ITEM(J+1) THEN GO TO S1;
 S2:
        TEMP= ITEM(J);
        ITEM(J) = ITEM(J+1);
        ITEM(J+1) = TEMP;
        ISWAP=1;
        GC TC S1;
  S1C:
        IF ISWAP =0 THEN GO TO S20;
        ISWAP=0;
        NUM = NUM -1;
        J=0;
        GC TC S1;
  S2C:
        PUT SKIP LIST('NO OF ITEMS SORTED=',N);
        RETURN;
  END SORT;
  ENDCON:
        ENC CEMO1;
/*
//GO.ASPEN  DD DSNAME=CONCAL,UNIT=TAPE9,VOLUME=SER=ILR080,
//   DCB=(RECFM=FB,BLKSIZE=1615,LRECL=85),DISP=OLD,LABEL=(1,NL,,IN)
//GO.LEX  DD  SYSOUT=A,DCB=(LRECL=20,RECFM=VBA,BLKSIZE=4000),
// SPACE=(TRK,40,RLSE)
//GO.SYSIN  DD  *
1
  ALL
WITHIN
OF
THAT
ON
A
FIVE
WAS
IS
BY
DURING
IN
```

Program Listing (Continued)

```
NOT
OR
BOTH
BEING
FOR
THE
HAVE
IF
NO
THEY
ONE
WERE
TWO
AND
BE
THIS
HAS
LESS
AS
HIM
SUCH
D
C
MAY
THEIR
OWN
ANY
SO
IT
UPON
SHALL
WHEN
WITH
WHICH
PRIOR
TO
AFTER
BECALSE
  AN
  BEFORE
SCME
MANY
TODAY
ITS
FROM
WHO
INTO
  OTHER
  AT
0
/*
//
```

9. Sample Output

CAL SIATS 0027 0000

A CAL WELFARE AND INSTITUTIONS CODE, SEC 18873

C THE DEPARTMENT OF EDUCATION MAY PROVIDE EMPLOYMENT OPPORTUNITIES FOR SUCH BLIND AND PHYSICALLY HANDICAPPED PERSONS WHO INDICATE A DESIRE TO CONTRIBUTE TO THEIR OWN SUPPORT THROUGH PARTIAL OR FULL-TIME EMPLOYMENT IN AN OPPORTUNITY WORK CENTER FOR THE BLIND, THROUGH PARTICIPATION IN THE PRODUCTION OF ARTICLES FOR SALE IN THE LOCAL COMMUNITY, THROUGH THE SALES FACILITIES OF CALIFORNIA INDUSTRIES FOR THE BLIND, OR THROUGH PARTICIPATION IN ASSEMBLY PROJECTS SECURED FOR THE OPPORTUNITY CENTER FROM PRIVATE BUSINESS ENTERPRISES OR FROM PUBLIC AGENCIES

D WELFARE - PUBLIC SOCIAL SERVICES - RELIEF LAW OF 1945

Dictionary Sort

NO OF ITEMS SORTED= 54

SORTED LIST

1	AGENCIES
1	ARTICLES
1	ASSEMBLY
3	BLIND
1	BUSINESS
1	CALIFORNIA
2	CENTER
1	COMMUNITY
1	CONTRIBUTE
1	DEPARTMENT
1	DESIRE
1	EDUCATION
2	EMPLOYMENT
1	ENTERPRISES
1	FACILITIES
1	FULL
1	HANDICAPPED
1	INDICATE
1	INDUSTRIES
1	LAW
1	LOCAL
1	OPPORTUNITIES
2	OPPORTUNITY
1	PARTIAL
2	PARTICIPATION
1	PERSONS
1	PHYSICALLY
1	PRIVATE
1	PRODUCTION
1	PROJECTS
1	PROVIDE
2	PUBLIC
1	RELIEF
1	SALE
1	SALES
1	SECURED
1	SERVICES
1	SOCIAL
1	SUPPORT
4	THROUGH
1	TIME
1	WELFARE
1	WORK
1	1945

Frequency Sort

NO OF ITEMS SORTED= 44

FREQUENCY LIST

4	THROUGH
3	BLIND
2	PUBLIC
2	PARTICIPATION
2	OPPORTUNITY
2	EMPLOYMENT
2	CENTER
1	1945
1	WORK
1	WELFARE
1	TIME
1	SUPPORT
1	SOCIAL
1	SERVICES
1	SECURED
1	SALES
1	SALE
1	RELIEF
1	PROVIDE
1	PROJECTS
1	PRODUCTION
1	PRIVATE
1	PHYSICALLY
1	PERSONS
1	PARTIAL
1	OPPORTUNITIES
1	LOCAL
1	LAW
1	INDUSTRIES
1	INDICATE
1	HANDICAPPED
1	FULL
1	FACILITIES
1	ENTERPRISES
1	EDUCATION
1	DESIRE
1	DEPARTMENT
1	CONTRIBUTE
1	COMMUNITY
1	CALIFORNIA
1	BUSINESS
1	ASSEMBLY
1	ARTICLES
1	AGENCIES

KWIC-type Record

C THE DEPARTMENT OF EDUCATION, WITH THE APPROVAL OF
C THE DEPARTMENT OF EDUCATION, WITH THE APPROVAL OF THE
C THE DEPARTMENT OF EDUCATION, WITH THE APPROVAL OF THE DIRECTOR O
C THE DEPARTMENT OF EDUCATION, WITH THE APPROVAL OF THE DIRECTOR OF G
C THE DEPARTMENT OF EDUCATION, WITH THE APPROVAL OF THE DIRECTOR OF GENERAL SER
C THE DEPARTMENT OF EDUCATION, WITH THE APPROVAL OF THE DIRECTOR OF GENERAL SERVICES,
C THE DEPARTMENT OF EDUCATION, WITH THE APPROVAL OF THE DIRECTOR OF GENERAL SERVICES, MAY
C THE DEPARTMENT OF EDUCATION, WITH THE APPROVAL OF THE DIRECTOR OF GENERAL SERVICES, MAY C (A)
C THE DEPARTMENT OF EDUCATION, WITH THE APPROVAL OF THE DIRECTOR OF GENERAL SERVICES, MAY C (A) RE
THE DEPARTMENT OF EDUCATION, WITH THE APPROVAL OF THE DIRECTOR OF GENERAL SERVICES, MAY C (A) RENT,
DEPARTMENT OF EDUCATION, WITH THE APPROVAL OF THE DIRECTOR OF GENERAL SERVICES, MAY C (A) RENT, LEASE, OR
T OF EDUCATION, WITH THE APPROVAL OF THE DIRECTOR OF GENERAL SERVICES, MAY C (A) RENT, LEASE, OR OT
F EDUCATION, WITH THE APPROVAL OF THE DIRECTOR OF GENERAL SERVICES, MAY C (A) RENT, LEASE, OR OTHERWISE
ION, WITH THE APPROVAL OF THE DIRECTOR OF GENERAL SERVICES, MAY C (A) RENT, LEASE, OR OTHERWISE PROVIDE S
THE APPROVAL OF THE DIRECTOR OF GENERAL SERVICES, MAY C (A) RENT, LEASE, OR OTHERWISE PROVIDE SUITAB
PROVAL OF THE DIRECTOR OF GENERAL SERVICES, MAY C (A) RENT, LEASE, OR OTHERWISE PROVIDE SUITABLE R
VAL OF THE DIRECTOR OF GENERAL SERVICES, MAY C (A) RENT, LEASE, OR OTHERWISE PROVIDE SUITABLE ROOM
OF THE DIRECTOR OF GENERAL SERVICES, MAY C (A) RENT, LEASE, OR OTHERWISE PROVIDE SUITABLE ROOMS OR BU
DIRECTOR OF GENERAL SERVICES, MAY C (A) RENT, LEASE, OR OTHERWISE PROVIDE SUITABLE ROOMS OR BUILDINGS
OR OF GENERAL SERVICES, MAY C (A) RENT, LEASE, OR OTHERWISE PROVIDE SUITABLE ROOMS OR BUILDINGS TO
OF GENERAL SERVICES, MAY C (A) RENT, LEASE, OR OTHERWISE PROVIDE SUITABLE ROOMS OR BUILDINGS TO HOUSE THE
SERVICES, MAY C (A) RENT, LEASE, OR OTHERWISE PROVIDE SUITABLE ROOMS OR BUILDINGS TO HOUSE THE ACTIVITI
S, MAY C (A) RENT, LEASE, OR OTHERWISE PROVIDE SUITABLE ROOMS OR BUILDINGS TO HOUSE THE ACTIVITIES OF THE
C (A) RENT, LEASE, OR OTHERWISE PROVIDE SUITABLE ROOMS OR BUILDINGS TO HOUSE THE ACTIVITIES OF THE OPPOR
RENT, LEASE, OR OTHERWISE PROVIDE SUITABLE ROOMS OR BUILDINGS TO HOUSE THE ACTIVITIES OF THE OPPORTUN
NT, LEASE, OR OTHERWISE PROVIDE SUITABLE ROOMS OR BUILDINGS TO HOUSE THE ACTIVITIES OF THE OPPORTUNITY CENTER
OR OTHERWISE PROVIDE SUITABLE ROOMS OR BUILDINGS TO HOUSE THE ACTIVITIES OF THE OPPORTUNITY CENTERS F
OTHERWISE PROVIDE SUITABLE ROOMS OR BUILDINGS TO HOUSE THE ACTIVITIES OF THE OPPORTUNITY CENTERS FOR THE
WISE PROVIDE SUITABLE ROOMS OR BUILDINGS TO HOUSE THE ACTIVITIES OF THE OPPORTUNITY CENTERS FOR THE BLI
PROVIDE SUITABLE ROOMS OR BUILDINGS TO HOUSE THE ACTIVITIES OF THE OPPORTUNITY CENTERS FOR THE BLIND C (B)
ITABLE ROOMS OR BUILDINGS TO HOUSE THE ACTIVITIES OF THE OPPORTUNITY CENTERS FOR THE BLIND C (B) E
BLE ROOMS OR BUILDINGS TO HOUSE THE ACTIVITIES OF THE OPPORTUNITY CENTERS FOR THE BLIND C (B) EMPLO
ROOMS OR BUILDINGS TO HOUSE THE ACTIVITIES OF THE OPPORTUNITY CENTERS FOR THE BLIND C (B) EMPLOY SUCH EMPLO
LDINGS TO HOUSE THE ACTIVITIES OF THE OPPORTUNITY CENTERS FOR THE BLIND C (B) EMPLOY SUCH EMPLOYEES AT
O HOUSE THE ACTIVITIES OF THE OPPORTUNITY CENTERS FOR THE BLIND) C (B) EMPLOY SUCH EMPLOYEES AT THE
USE THE ACTIVITIES OF THE OPPORTUNITY CENTERS FOR THE BLIND) C (B) EMPLOY SUCH EMPLOYEES AT THE CENT
THE ACTIVITIES OF THE OPPORTUNITY CENTERS FOR THE BLIND C (B) EMPLOY SUCH EMPLOYEES AT THE CENTERS AS
ITIES OF THE OPPORTUNITY CENTERS FOR THE BLIND C (B) EMPLOY SUCH EMPLOYEES AT THE CENTERS AS IT D
ES OF THE OPPORTUNITY CENTERS FOR THE BLIND C (B) EMPLOY SUCH EMPLOYEES AT THE CENTERS AS IT DEEM
E THE OPPORTUNITY CENTERS FOR THE BLIND C (B) EMPLOY SUCH EMPLOYEES AT THE CENTERS AS IT DEEMS NECESSA
PPORTUNITY CENTERS FOR THE BLIND C (B) EMPLOY SUCH EMPLOYEES AT THE CENTERS AS IT DEEMS NECESSARY.
UNITY CENTERS FOR THE BLIND C (B) EMPLOY SUCH EMPLOYEES AT THE CENTERS AS IT DEEMS NECESSARY. C (C) PE
ERS FOR THE BLIND C (B) EMPLOY SUCH EMPLOYEES AT THE CENTERS AS IT DEEMS NECESSARY C (C) PERFO
FOR THE BLIND C (B) EMPLOY SUCH EMPLOYEES AT THE CENTERS AS IT DEEMS NECESSARY. C (C) PERFORM A
THE BLIND C (B) EMPLOY SUCH EMPLOYEES AT THE CENTERS AS IT DEEMS NECESSARY C (C) PERFORM ALL NECES
ND C (B) EMPLOY SUCH EMPLOYEES AT THE CENTERS AS IT DEEMS NECESSARY C (C) PERFORM ALL NECESSAR
C (B) EMPLOY SUCH EMPLOYEES AT THE CENTERS AS IT DEEMS NECESSARY C (C) PERFORM ALL NECESSARY A
C (B) EMPLOY SUCH EMPLOYEES AT THE CENTERS AS IT DEEMS NECESSARY C (C) PERFORM ALL NECESSARY AND PRO
EMPLOY SUCH EMPLOYEES AT THE CENTERS AS IT DEEMS NECESSARY C (C) PERFORM ALL NECESSARY AND PROPER ACTS T
EMPLOYEES AT THE CENTERS AS IT DEEMS NECESSARY C (C) PERFORM ALL NECESSARY AND PROPER ACTS TO CAR
LOYEES AT THE CENTERS AS IT DEEMS NECESSARY C (C) PERFORM ALL NECESSARY AND PROPER ACTS TO CARRY
ES AT THE CENTERS AS IT DEEMS NECESSARY C (C) PERFORM ALL NECESSARY AND PROPER ACTS TO CARRY OUT THE PR
E CENTERS AS IT DEEMS NECESSARY C (C) PERFORM ALL NECESSARY AND PROPER ACTS TO CARRY OUT THE PROVIS
NTERS AS IT DEEMS NECESSARY C (C) PERFORM ALL NECESSARY AND PROPER ACTS TO CARRY OUT THE PROVISIONS OF TH
T DEEMS NECESSARY C (C) PERFORM ALL NECESSARY AND PROPER ACTS TO CARRY OUT THE PROVISIONS OF THIS A
SSARY C (C) PERFORM ALL NECESSARY AND PROPER ACTS TO CARRY OUT THE PROVISIONS OF THIS ARTICLE
ESSARY C (C) PERFORM ALL NECESSARY AND PROPER ACTS TO CARRY OUT THE PROVISIONS OF THIS ARTICLE D
Y C (C) PERFORM ALL NECESSARY AND PROPER ACTS TO CARRY OUT THE PROVISIONS OF THIS ARTICLE D WEL
C (C) PERFORM ALL NECESSARY AND PROPER ACTS TO CARRY OUT THE PROVISIONS OF THIS ARTICLE. D WELFARE -

Moore Business Forms, Inc.

III. PROGRAM TESTMT

1. Identification

Title	TESTMT
User Group	Institute of Library Research
Category	CIS Text Processing
Source Language	PL/1
Machine Configuration	IBM/360 Model 91, a tape, printer
Space Requirements	Step Region:
	PL1L 162K
	LKED 154K
	GO 250K
	Core Used 178K
Author	G. Silva
Date	March 1970

2. Purpose

This program, like program DEMO1, also exemplifies some of the basic procedures employed in most language-data processing systems. However, it has some additional capabilities, and the techniques used are somewhat more elegant. For example, the program uses the preprocessor, pointer variables, and a binary tree sort. The capabilities of program TESTMT are listed below:

1. It reads textual material from magnetic tape.
2. It examines the text character by character, and, for each incoming character it prints out its decimal value, its binary value, and its hexadecimal value.

3. It performs a table lookup and prints out the substitute value for each character.
4. It searches the text for given terms (these may be multiple word phrases or uniterms), and prints out those found with a given amount of context on each side.
5. It performs a dictionary sort on the words of the text and prints out the results in the form of a word frequency list.

The above operations are carried out in internal memory, and the amount of text processed at any given time is still limited.

The Aspen file, which contains full-text images of documents from the California Code and State Constitution, was used for testing purposes.

3. Input Specifications

Input is accepted from the Aspen tape which has the following specifications:

1. 9 track tape
2. 800 BPI
3. No header or trailer labels
4. First file on tape ILR080
5. Volume serial number is ILR080
6. Data set name is CONCAL
7. Record format is fixed blocked (FB)
8. Logical record length is 85 bytes
9. Block size is 1615

The DD card used to read the tape is:

```
//GO.ASPEN  DD DSNAME=CONCAL,UNIT=TAPE9,VOLUME=SER=ILR080,
//  DCB=(RECFM=FB,BLKSIZE=1615,LRECL=85),DISP=OLD,LABEL=(1,NL,,IN)
```

4. File Description

The file contains fixed blocked records throughout with the following record format:

Positions 1 - 5, ascending document numbers,

Positions 6 - 85, text.

The first record of each document contains indicative data for that document instead of text.

5. Checkout

5.1 Timing

The program processed the first three documents from the Aspen tape. The documents contained a total of 222 words before exclusion, and 118 after exclusion.

The CPU time for the various steps is set out below:

Step Name	Step CPU Seconds
PL1L	12.98s
LKED	1.28s
GO	3.92s

Average processing time=2.3s/100 input words

5.2 Error Checking

None

5.3 Cost

The cost of running this program was:

PL1L	$7.22
LKED	2.55
GO	1.85
Total	$11.62

5.4 Remarks

It is worth noting that the use of the preprocessor results in a large increase in compile time. It is therefore debatable whether the convenience afforded by the use of the preprocessor is worth the extra expense.

The amount of text that can be processed in core is limited by the size of array BP which holds the words sorted in subroutine TREE.

Due to PL/1 restrictions upon the length of character strings, the maximum size allowable would be approximately 1600x20 bytes. Assuming the average word length to be 5 characters, and allowing for one blank between words, the maximum permissible length for one section of text is approximately 9600 bytes.

6. Program Description

The program consists of a main program and various autonomous subroutines, each corresponding to one of the main functions set out in paragraph 2.

Main Steps

The program carries out the following operations in the order indicated below:

1. The preprocessor stage

   Here the user may exert control over his source program, change identifiers appearing in the source program, and indicate which sections of the program are to be compiled. The following options are under the user's control at compile-time.

Switches

DEBUG=1 will include a number of debugging routines

EXCLUDE=1 will cause the preprocessor to compile the various program sections needed when an exclusion list is used

SYMCON=0 excludes subroutine SYMCONV and TABLES

HEXCON=0 excludes subroutine HEXCONV

Identifiers

EXCL_NO maximum number of items allowable in exclusion list (presently set at 100)

NPHRASES maximum number of phrases permissible (presently set at 20)

CONTEXT length of context desired (presently set at 50)

LLINE length of output line in subroutine LOOKUP (presently set at 140)

2. The processor stage

a. All variables are declared and initialized.

b. The program calls subroutine TABLES, which reads in the tables required for the conversion from the source character encoding scheme to the one appropriate for use on the present installation.

c. The program calls subroutine READIN which reads into memory the index terms supplied by the user, to be used for retrieval in subroutine LOOKUP. Both the conversion table and the index terms are read in from cards.

d. The program reads the Aspen file and stores the text of one document at a time in the character array STATUTE.

e. In the next step subroutine HEXCONV produces a hexadecimal version of the whole text. This operation is very useful when one is not certain of the content of a new file and one wishes to check the structure of the record, for example.

f. The next call is to subroutine SYMCONV, which examines the input text character by character, performs the required character conversion by table lookup, and prints out for each character its decimal value, its binary value, and its hexadecimal value. This subroutine was designed for demonstration purposes.

g. Next subroutine LOOKUP retrieves the index terms within context, and signals when a search was not successful.

h. Lastly, subroutine TREE performs a dictionary sort in the words of the text, and prints them out both in dictionary order, and in order of decreasing frequency.

7. Program Listing

```
//NRB409GR  JOB  '20,2500',SILVA
// EXEC PL1LFCLG,PARM='M,SM=(2,72,1),ST',PG=200,RG=250K
//PL1L.SYSLIN  DD  SPACE=(TRK,(20,10))
//PL1L.SYSUT1  DD  SPACE=(TRK,(20,10))
//PL1L.SYSUT3  DD  SPACE=(TRK,(20,10))
//PL1L.SYSIN DD *
  TESTMT:
         PROC OPTIONS(MAIN);
         % DCL DEBUG FIXED;
         % DEBUG =1;
         % DEBUG = 0;
 % DCL HEXCON FIXED;
 % HEXCON = 0;
 % HEXCON = 1;
 % DCL SYMCON FIXED;
 % SYMCON = 0;
 % SYMCON = 1;
         % DCL NPHRASES FIXED;
         % NPHRASES = 20;
         % DCL EXCLUDE FIXED;
         % EXCLUDE =0;
         % EXCLUDE =1;
         % DCL EXCL_NO FIXED;
         % EXCL_NO = 100;
         DCL REC CHAR(85)VAR,
         ULINE CHAR(80)VAR INIT((80)'_'),
           (SEQID1,SEQID2)CHAR(5) INIT('00001');
         DCL STATUTE CHAR(32500)VAR;
/* ALL LISTS OF ALPHABETIC ITEMS TO BE READ IN FROM CARDS
   MUST BE DECLARED CHAR(82)VAR. THIS REQUIREMENT IS
   IMPOSED BY THE NORMALIZING PROCEDURE IN  "READIN"  */
         % IF EXCLUDE=0 % THEN % GO TO EX05;
         DCL EXCL_LIST(EXCL_NO) CHAR(82)VAR;
% EX05:;
/* PHRASE CONTAINS THE LIST OF INDEX TERMS  */
         DCL PHRASE (NPHRASES) CHAR(82)VAR;
         DECLARE(CHARTAB(125), NOTAB(20))FIXED;

/*  IRD COUNTS NUMBER OF DATA SETS READ IN FROM CARDS  */
         IRD=0;
% IF SYMCON=0 % THEN % GO TO DB09;
/*  THIS SECTION READS CARD INPUT  */
         PUT SKIP LIST('CHARACTER CONVERSION TABLE');
         CALL TABLES;
         IRD = IRD +1;
% DB09:;
         IF IRD >0 THEN GET SKIP;
/* READ IN LIST OF INDEX TERMS  */
         PUT SKIP (5) LIST('INDEX TERMS');
         CALL READIN(PHRASE,NO_PHRASES);
         DO J=1 TO NO_PHRASES;
         PUT SKIP LIST(PHRASE(J));
         END;
         IRD = IRD +1;
% IF EXCLUDE=0 % THEN % GO TO EX10;
/* READ IN EXCLUSION LIST  */
         IF IRD >0 THEN GET SKIP;
         PUT PAGE;
         PUT SKIP LIST(' EXCLUSION LIST');
         CALL READIN(EXCL_LIST,NO_EXCL);
/*  SORT EXCLUSION LIST INTO ALPHABETICAL ORDER
```

Program Listing (Continued)

```
   REQUIREMENT IMPOSED BY SUBSEQUENT BINARY SEARCH  */
        CALL SORT( NO_EXCL,EXCL_LIST);
 % EX10:;
        ON ENDFILE(ASPEN) GO TO FINIS;
/*  BEGINNING OF FIRST STATUTE */
        PUT PAGE;
        NREQ = 1;
        KT = 1;
        READ FILE(ASPEN) INTO (REC);
        GO TO HO5;
/* PROCESS HEADING AND SUBHEADING */
HO2:
        READ FILE(ASPEN) INTO (REC);
        KT = KT + 1;
HO5:
        REC = SUBSTR(REC, 6);
        LEN = 81;
LOOP1:
        LEN = LEN-1;
        IF SUBSTR(REC, LEN, 1) = ' ' THEN GO TO LOOP1;
/*  LEN EQUALS LENGTH OF REC  */
        PUT EDIT(REC)(SKIP(2), COL(30),A);
        PUT EDIT(ULINE)(SKIP(0), COL(30), A(LEN));
        IF KT<2 THEN GO TO HO2;
/*  BEGIN PROCESSING TEXT OF STATUTE */
        STATUTE = '';
        SEQID1 = SEQID2;
NEXTLIN:
        READ FILE(ASPEN) INTO (REC);
        SEQID2 = SUBSTR(REC, 1, 5);
        IF SEQID2>SEQID1 THEN DO;
        NREQ = NREQ+1;
        KT = 1;
        GO TO IND;
        END;
S10:
        REC = SUBSTR(REC, 6);
        PUT SKIP LIST(REC);  /* PRINTS NEXT LINE OF TEXT  */
        DO K=80 TO 1 BY -1 WHILE (SUBSTR(REC, K, 1) = ' ');
        END;
        K = K+1;
        REC = SUBSTR(REC, 1, K);
        STATUTE = STATUTE || REC;
        GO TO NEXTLIN;
 IND:
% IF SYMCON=0 % THEN % GO TO DB15;
        PUT SKIP(5) LIST(' SUBROUTINE SYMCON');
        PUT EDIT(ULINE)(SKIP(0), A (18));
        CALL SYMCONV(STATUTE);
 % DB15:;
        PUT SKIP(2) LIST('STATUTE', STATUTE);
 % IF HEXCON=0 % THEN % GO TO DB20;
        PUT SKIP(5) LIST(' SUBROUTINE HEXCON');
        PUT EDIT(ULINE)(SKIP(0), A (18));
        CALL HEXCONV(STATUTE);
% DB20:;
-
 /* RETRIEVE  INDEX TERMS  */
        PUT SKIP(5) LIST(' SUBROUTINE LOOKUP',
'INDEX TERMS IN CONTEXT');
        PUT EDIT(ULINE)(SKIP(0), A (47));
```

Program Listing (Continued)

```
        CALL LOOKUP(STATUTE, NO_PHRASES, PHRASE);
        CALL TREE(STATUTE, EXCL_LIST, NO_EXCL);
        IF NREQ>3 THEN GO TO FINIS;
        PUT PAGE;
        GO TO HO5;
% IF HEXCON=0 % THEN % GO TO DB25;
 HEXCONV:
        PROC(TEXT);
        DCL TEXT CHAR(*);
        DCL TEXT_KAR CHAR(1)BASED(P);
        DCL HEX(0:15) CHAR(1) INIT('0','1','2','3','4','5','6','7',
        '8','9','A','B','C','D','E','F');
        DECLARE BIN BIT(8) VAR;
        DCL SS CHAR(602)VAR;
        DCL P POINTER;
        DCL PROXY FIXED BIN(31)BASED(PROXY_PTR);
        SS= 'SS';
        PROXY_PTR=ADDR(P);
        P= ADDR(TEXT);
        DO I=1 TO 600;
        BIN = UNSPEC(TEXT_KAR);
        SS = SS ||HEX(SUBSTR(BIN,1,4))||HEX(SUBSTR(BIN,5,4));
        PROXY=PROXY+1;
        END;
        PUT SKIP(2) LIST( SS);
        RETURN;
 END HEXCONV;
% DB25:;
% IF SYMCON=0 % THEN % GO TO DB30;
 SYMCONV:
        PROCEDURE(INTEXT);
        DCL INTEXT CHAR(*)VAR;
        DCL DEC FIXED BINARY(8,0);
        DECLARE BIN BIT(8) VAR;
        DCL HEX(0:15) CHAR(1) INIT('0','1','2','3','4','5','6','7',
        '8','9','A','B','C','D','E','F');
        DCL HEX_KAR CHAR(2);
        DCL NEW_KAR CHAR(1);
        DCL TEXT_KAR CHAR(1);
        LENGTH_INTEXT = LENGTH(INTEXT);
        KOUNT=0;
NEXTKAR:
        KOUNT=KOUNT+1;
        IF KOUNT > LENGTH_INTEXT THEN RETURN;
        TEXT_KAR = SUBSTR(INTEXT, KOUNT, 1);
 /* TRANSFORM INPUT CHARACTER INTO BIT STRING    */
        BIN = UNSPEC ( TEXT_KAR);
 /*    FIND DECIMAL VALUE OF BIN    */
        DEC = BIN;
 /* GET HEXADECIMAL VALUE OF BIN       */
        HEX_KAR = HEX(SUBSTR(BIN,1,4))||HEX(SUBSTR(BIN,5,4));
 /* LOOK UP CHARACTER IN NUMBER-TABLE    */
        IF DEC>239 THEN DO;
        DEC = DEC-239;
        ICH = NOTAB(DEC);
            GO TO PRINT1;
                              END;
 /* IF CHARACTER NOT ALPHA-NUMERIC      */
        IF DEC<193 THEN DO;
        SUBSTR(INTEXT, KOUNT, 1) = TEXT_KAR;
        GO TO NEXTKAR;
```

Program Listing (Continued)

```
                              END;
        DEC = DEC-192;
/* LOOK UP CHARACTER IN ALPHA-TABLE    */
        ICH = CHARTAB(DEC);
PRINT1:
/* THIS CONVERTS THE DECIMAL VALUE OF THE NUMBER LOOKED UP
 INTO CHARACTER REPRESENTATION  */
        UNSPEC(NEW_KAR)= SUBSTR(UNSPEC(ICH),25,8);
        SUBSTR(INTEXT, KOUNT, 1) = NEW_KAR;
/*  FOR DEMONSTRATION  PURPOSES ONLY  */
/* PRINT OUT NO MORE THAN 30 CHARACTERS  */
        IF KOUNT>30 THEN GO TO NEXTKAR;
        PUT EDIT( 'INPUT CHARACTER=', TEXT_KAR, 'DEC_VALUE=', DEC,
        'BIN= ',
        BIN, 'HEX_VALUE =', HEX_KAR, 'NEW_KAR= ', NEW_KAR)
        (X(2), A(16), A(1), X(4), A(10), F(3), X(4), A(5), B(8), X(4),
        A(11), A(2), X(4), A(9), A(2))SKIP(3);
        GO TO NEXTKAR;
END SYMCONV;
% DB30:;
LOOKUP:
        PROC(TEXT, NO_PHRASES, PHRASE);
/* THIS SUBROUTINE RETRIEVES A GIVEN NUMBER OF TERMS
        FROM TEXT AND PRINTS EACH TERM IN CONTEXT    */
        % DCL LLINE FIXED;
        % LLINE = 140;
        % DCL CONTEXT FIXED;
        % CONTEXT=50;
        DCL TEXT CHAR(*)VAR;
        DCL (FRONT,BACK)CHAR(CONTEXT)VAR;
        DCL OUTPUT CHAR(LLINE)VAR;
        DCL ( I , B , POINT) FIXED BIN;
        DCL N FIXED BIN;
        DCL ULINE CHAR(50) INIT((50)'_');
        DCL NO_PHRASES BIN FIXED;
        DCL PHRASE(*) CHAR(*)VAR;
        LTEXT=LENGTH(TEXT);
        I=0;
NEXTPHR:
        I=I+1;
        IF I> NO_PHRASES THEN RETURN;
        POINT=1;
        FRONT= (CONTEXT)' ';
        BACK = (CONTEXT)' ';
        LENGTH_PHRASE= LENGTH(PHRASE(I));
        PUT SKIP(2) LIST(PHRASE(I));
        N=0;
SEARCH:
        N=N+1;
        B = INDEX(SUBSTR(  TEXT  ,POINT),PHRASE(I));
STEP1:
/*     PHRASE NOT FOUND IN TEXT     */
        IF B =0 THEN DO;
        IF N=1 THEN DO;
/* IF B=0 WHEN N=1 THE SEARCH WAS UNSUCCESSFUL    */
                        PUT SKIP LIST('SEARCH UNSUCCESSFUL');
                        GO TO NEXTPHR;
        END;
        GO TC NEXTPHR;
                    END;
STEP2:
```

Program Listing (Continued)

```
        IF B+POINT-2 < CONTEXT THEN DO;
            FRONT = SUBSTR(  TEXT  , POINT  , B-1);
            GO TO STEP3;
                                        END;
          FRONT = SUBSTR(   TEXT  , B-CONTEXT+POINT-1, CONTEXT);
STEP3:
        IF LTEXT            -B-LENGTH_PHRASE-POINT+2 < CONTEXT THEN DO;
            BACK =  SUBSTR(   TEXT  , B+ LENGTH_PHRASE+POINT-1);
            GO TO STEP4;
                                                              END;
          BACK = SUBSTR(   TEXT  , B+LENGTH_PHRASE +POINT-1, CONTEXT);
STEP4:
        OUTPUT= FRONT||PHRASE(I)||BACK;
        POINT = POINT + B + LENGTH_PHRASE -1;
PRINT2:
        PUT SKIP EDIT(OUTPUT)(A);
        PUT EDIT(ULINE)(SKIP(0),COL(51), A(LENGTH_PHRASE));
        GO TO SEARCH;
END LOOKUP;

READIN:
        PROC(LINE,NUM);
/*   READS IN AND PRINTS A LIST OF TERMS TYPED ONE PER CARD    */
/*  A "TERM" MAY CONSIST OF ONE WORD OR SEVERAL WORDS
    SEPARATED BY AT LEAST ONE BLANK   */
/* DATA DECK MUST END WITH A ZERO IN COL 1  */
/* NO OTHER FORMAT RESTRICTIONS ARE IMPOSED  */
        DCL LINE (*)CHAR(*)VAR;
        DCL (N,I) FIXED BIN;
        N=0;
R10:
        N=N+1;
        GET EDIT(LINE(N)) (A(80));
        IF LINE(N) = '0' THEN GO TO R50;
/* NORMALIZE CARD FORMAT  */
/* ADD BLANK AT THE BEGINNING AND AT THE END OF  EACH LINE  */.
        LINE(N) = ' '||LINE(N)||' ';
/* REDUCE MULTIPLE BLANKS TO ONE BLANK  */
R20:
        I = INDEX(LINE(N),'  ');
        IF I¬=0 THEN DO ;
            IF I=LENGTH(LINE(N))-1  THEN DO ;
            LINE(N) = SUBSTR(LINE(N),1,I);
            GO TO R30;
            END;
        LINE(N) = SUBSTR(LINE(N),1,I)||SUBSTR(LINE(N),I+2);
        GO TO R20;
        END;
/* REMOVE ALL BLANKS BEFORE AND AFTER LINE  */
R30:
        LINE(N) = SUBSTR(LINE(N),2,LENGTH(LINE(N))-2);
        GO TC R10;
R50:
        NUM = N-1;
        RETURN;
END READIN;

SORT:
        PROC(N,ITEM);
        DCL TEMP CHAR(80)VAR;
        DCL ITEM(*) CHAR(*) VAR;
```

Program Listing (Continued)

```
        NUM = N-1;
        J=0;
S1:     J=J+1;
        IF J>NUM THEN GO TO S10;
        IF ITEM(J) > ITEM(J+1) THEN DO;
S2:
        TEMP= ITEM(J);
        ITEM(J) = ITEM(J+1);
        ITEM(J+1) = TEMP;
        ISWAP=1;
        GO TO S1;
                        END;
        GO TO S1;
S10:
        IF ISWAP =0 THEN GO TO S20;
        ISWAP=0;
        NUM = NUM -1;
        J=0;
        GO TO S1;
S20:
        DO J=1 TO N;
          PUT SKIP LIST(ITEM(J));
        END;
        PUT SKIP (2) LIST(' NO OF ITEMS SORTED = ',N);
        RETURN;
END SORT;
-
  % IF SYMCON=0 % THEN % GO TO DB40;
  TABLES:  PROC;
         DCL I FIXED BIN;
         CHARTAB=0;
         NOTAB=0;
  /*  READ IN ALPHA-TABLE        */
         I=0;
  READ1:
         I=I+1;
         GET LIST(CHARTAB(I));
         IF CHARTAB(I)¬=999 THEN GO TO READ1;
  /*   READ IN NUMBER-TABLE*/
         I=0;
  READ2:
          I=I+1 ;
         GET LIST( NOTAB(I));
         IF NOTAB(I) ¬= 999 THEN GO TO READ2;
  /*    PRINT ALPHA-TABLE    */
         PUT SKIP LIST (CHARTAB);
  /*    PRINT NUMBER-TABLE    */
         PUT SKIP LIST (NOTAB);
         RETURN;
  END TABLES;
 % DB40:;
-
 TREE: PROCEDURE(IP, EXCL_LIST, NO_EXCL);
         % DCL WORDL FIXED;
         % WORDL=20;
         % DCL NOW FIXED;
         % NOW = 500;
         % DCL FB CHAR;
         % FB='FIXED BINARY';
    /* FOR REJECTION OF ITEMS EXCLUDE=1, FOR NO REJECTION EXCLUDE=0  */
         DCL EXCL_LIST(EXCL_NO)CHAR(82)VAR;
```

Program Listing (Continued)

```
/* EXCL_LIST CONTAINS ITEMS TO BE EXCLUDED FROM THE TREE SORT.
  ALL LISTS OF ALPHA_ITEMS TO BE READ IN FROM CARDS MUST BE
  DECLARED " CHAR(82)VAR ". THIS REQUIREMENT IS IMPOSED BY THE
 NORMALIZING PROCEDURE IN  " READIN"  */
       DCL COMPARE RETURNS(FIXED BIN);
% IF DEBUG=0 % THEN % GO TO DB50;
       DCL TALES PRINT;
% DB50:;
       DCL I FB;
      DCL  HEXD RETURNS(CHAR(8));
       DCL IP CHAR(*)VAR;
DCL JEOF FB INIT(0);
  DCL  (A,B,C) PTR;
  DCL  NI   CHAR (WORDL);
       DCL (FIRST, LAST) BIN FIXED;
  DCL  (ITEM,HEAD)  PTR;
 DCL MAXLIP FB;
       DCL TP PTR;
DCL   HED  PTR;
       DCL 1 THING BASED(TP),
             2 COUNT FB,
             2 NAME PTR,
             2 LEF PTR,
             2 RIT PTR;
       DCL BP(NOW) CHAR(WORDL);
      ON ERROR SNAP BEGIN;
                  CALL  IHEDUMP;
                  END;
        HED = NULL;
        JCOMP =1;
        MAXLIP = LENGTH(IP);
% IF DEBUG=0 % THEN % GO TO DB60;
       PUT FILE(TALES) PAGE LIST ('MAXLIP',MAXLIP,'IP',IP);
% DB60:;
         I=1;
  HEAD =NULL;
   DO   J=1 BY 1 TO NOW   WHILE (I  <= MAXLIP );
/*   NI CONTAINS THE NEXT ITEM  */
T20:
/* IT LOOPS WHILE BLANKS OR PUNCTUATION IS ENCOUNTERED */
       DO FIRST = I TO MAXLIP WHILE
       (INDEX(' ,.:?"=()-', SUBSTR(IP   , FIRST, 1)) ¬= 0);
       END;
/* IT LOOPS WHILE NO PUNCTUATION IS ENCOUNTERED */
       DO LAST = FIRST+1 TO MAXLIP WHILE
       (INDEX(' ,.:?"=()-', SUBSTR(IP   , LAST, 1)) = 0);
       END;
/* NI IS THE NEXT ITEM */
       NI = SUBSTR(IP     , FIRST, (LAST-FIRST));
       I = LAST + 1;
% IF EXCLUDE=0 % THEN % GO TO EX20;
       CALL BIS(EXCL_LIST,NO_EXCL,NI,JFOUND,MID);
       IF JFOUND=1 THEN GO TO T20;
% EX20:;
      BP(J) = NI;
       ITEM= ADDR(BP(J));
% IF DEBUG=0 % THEN % GO TO DB70;
       PUT FILE(TALES) SKIP LIST(J,NI);
% DB70:;
   CALL  ADD  (HEAD,ITEM);
    END ;
```

Program Listing (Continued)

```
% IF DEBUG=0 % THEN % GO TO DB72;
        PUT SKIP LIST('LISTPRC(HEAD)');
        CALL LISTPRC(HEAD);
        PUT PAGE;
        PUT SKIP LIST('LISTRC(HEAD)');
        CALL LISTRC(HEAD);
        PUT PAGE;
        PUT SKIP LIST('LINPR(HEAD)');
        CALL LINPR(HEAD);
% DB72:;
        A,B,C=NULL;
        CALL LINERC(HEAD,A,B,C);
        PUT PAGE;
% IF DEBUG=0 % THEN % GO TO DB74;
        PUT SKIP LIST('LISTRC(HEAD)');
        CALL LISTRC(HEAD);
        PUT PAGE;
% DB74:;
        PUT SKIP LIST('LISTPRC(HEAD)');
        PUT SKIP LIST('  DICTIONARY SORT');
        PUT SKIP;
        CALL LISTPRC(HEAD);
        JCOMP = 2;
        CALL REORDER( HED, HEAD );
        CALL ADJUST(HED);
% IF DEBUG=0 % THEN% GO TO DB75;
        PUT PAGE;
        PUT SKIP LIST('LISTPRC(HED)');
        CALL LISTPRC(HED);
% DB75:;
        A,B,C = NULL;
        CALL LINERC(HED,A,B,C);
        PUT PAGE;
% IF DEBUG=0 % THEN % GO TO DB76;
        PUT SKIP LIST('LINPR(HED)');
        CALL LINPR(HED);
        PUT PAGE;
% DB76:;
        PUT SKIP LIST('LISTPRC(HED)');
        PUT SKIP LIST(' FREQUENCY LIST');
        PUT SKIP;
        CALL LISTPRC(HED);
        RETURN;
-
-
-
ADD:  PROC (HEAD,ITEM);
      DCL  HEAD  PTR;
      DCL  (OLD,TP) PTR;
      DCL  1 THING BASED (TP),
                2 COUNT FIXED BIN,
                2 NAME PTR,
                2 LEF PTR,
                2 RIT PTR;
      DCL  ITEM  PTR;
        DCL I FIXED BIN;
      TP = HEAD;
      DO  WHILE (TP¬=NULL);
      I = COMPARE (ITEM,NAME);
      IF I = 0 THEN DO;
                    COUNT=COUNT+1;
```

Program Listing (Continued)

```
                        RETURN;
                        END;
     OLD = TP;
     IF I < 0 THEN TP=LEF;
                 ELSE TP=RIT;
     END;
     ALLOCATE  THING  SET (TP);
/*  FOR DEBUGGING PURPOSES ONLY  */
% IF DEBUG=0 % THEN % GO TO DB80;
  CALL  TELL  ('THING',TP,'L');
% DB80:;
     NAME = ITEM;
     LEF, RIT = NULL;
     COUNT=1;
     IF HEAD = NULL THEN HEAD = TP;
                    ELSE IF I < 0  THEN OLD-> LEF = TP;
                                   ELSE OLD-> RIT = TP;
     RETURN;
END ADD;

ADJUST:
        PROC (TP)RECURSIVE;
        DCL TP PTR;
        DCL PP PTR;
        DCL 1 THING BASED(TP),
              2 COUNT FB,
              2 NAME PTR,
              2 LEF PTR,
              2 RIT PTR;
        IF TP = NULL THEN RETURN;
        IF LEF ¬= NULL THEN CALL ADJUST(LEF);
        PP = NAME;
        COUNT = PP->COUNT;
        NAME = PP -> NAME;
        IF RIT ¬= NULL THEN CALL ADJUST(RIT);
        RETURN;
END ADJUST;

% IF  EXCLUDE=0 % THEN % GO TO DB85;
BIS:
        PROC( LIST, LENGTH, ITEM, JFOUND, MID);
        /*  BIS RETURNS JFOUND=1 IF ELEMENT FOUND, ZERO, IF NOT  */
        DCL LIST(*        ) CHARACTER(* )VAR;
        DCL ITEM CHAR(*)VAR;
        LOWER=1;
        IUPPER = LENGTH +1;
B10:
        MID = (LOWER + IUPPER)/2;
        IF ITEM = LIST(MID) THEN DO;
        JFOUND=1;
        RETURN;
        END;
        IF ITEM > LIST(MID) THEN DO;
            IF(MID = LOWER)|(MID = IUPPER ) THEN DO;
            JFOUND=0;
            RETURN;
            END;
        LOWER = MID;
        GO TO B10;
                                END;
            IF (MID = LOWER)| ( MID = IUPPER) THEN DO;
```

Program Listing (Continued)

```
            JFOUND=0;
            RETURN;
            END;
         IUPPER=MID;
         GO TO B10;
  END BIS;
  % DB85:;
-
-
    COMPARE: PROC(IT,NM) FIXED BIN;
 /* THIS ROUTINE IS REPRESENTATION DEPENDENT  */
 /*
    RETURNS
      0          IF THE   TWO    ELEMENTS  ARE  EQUAL
      1          IF  THE FIRST COMES AFTER  THE SECOND
     -1          IF  THE FIRST COMES BEFORE THE SECOND
*/
          DCL  IT PTR;
     DCL  NM  PTR;
      DCL  ITEM CHAR(WORDL)  BASED (IT);
       DCL  1 THING BASED (IT),
                  2 COUNT FB,
                  2 NAME PTR,
                  2 LEF PTR,
                  2 RIT PTR;
        IF  JCOMP ¬=1 THEN GO TO S10;
   IF IT=NM THEN RETURN(0);
     IF NM->ITEM=ITEM THEN RETURN(0);
     IF NM->ITEM<ITEM THEN RETURN(1);
                      ELSE RETURN(-1);
  S10:
        IF COUNT <= NM->COUNT THEN RETURN(1);
                   ELSE RETURN (-1);
  END COMPARE;
-
-
  HEXD: PROC(PT) CHAR (8)  ;
    DCL BIN BIT(32);
    DCL  BA(8)  BIT(4)  DEF  BIN;
  DCL  HX CHAR (16) INIT('0123456789ABCDEF');
   DCL  HEX (0 : 15) CHAR (1) DEF HX;
    DCL  I  FB;
  DCL  PCHT CHAR (8);
  DCL  PCHA (8)  CHAR (1)  DEF PCHT;
      BIN=UNSPEC(PT);
    DO  I=1 TO  8;
    PCHA(I)=HEX(BA(I));
        END ;
    RETURN (PCHT);
  END HEXD;
-
 LINERC:  PROC(HEAD,TP,OLD,ALF) RECURSIVE;
 /*  LINEARISES TREE  */
        DCL (A,B)PTR;
      DCL (HEAD,OLD,ALF,TP) PTR;
      DCL  1 THING BASED (TP),
                 2 COUNT FIXED BIN,
                 2 NAME PTR,
                 2 LEF PTR,
                 2 RIT PTR;
      DCL LINK PTR BASED(ALF);
```

Program Listing (Continued)

```
      IF HEAD=NULL THEN RETURN;
      IF TP=NULL THEN TP=HEAD;
DOWN:
      A=LEF;
      B=TP;
      IF LEF¬=NULL THEN CALL LINERC(HEAD,A,B,ALF);
      IF ALF=NULL THEN HEAD=TP;
                  ELSE LINK=TP;
      ALF=ADDR(RIT);
      IF OLD ¬= NULL THEN OLD -> LEF=RIT;
      IF RIT=NULL THEN RETURN;
      TP=RIT;
      GO TO DOWN;
 END LINERC;
-
% IF DEBUG=0 % THEN % GO TO DB90;
 LINPR: PROC (TP) RECURSIVE;
 /* PRINTS TREE LINEARLY WITHOUT CHANGING STRUCTURE  */
      DCL  TP PTR;
      DCL 1 THING BASED(TP),
                  2 COUNT FIXED BIN,
                  2 NAME PTR,
                  2 LEF PTR,
                  2 RIT PTR;
   IF TP=NULL THEN RETURN;
  IF LEF¬=NULL THEN CALL LINPR(LEF);
      CALL LISTI(TP,0B);
   IF RIT¬=NULL THEN CALL LINPR(RIT);
      RETURN;
  END LINPR;
-
 % DB90:;
   LISTI: PROC(IT,I);
 /*  PRINTS ONE ITEM AT A TIME  */
 /* THIS ROUTINE IS REPRESENTATION DEPENDANT  */
 DCL  WORK CHAR(WORDL+7)    VAR;
  DCL  IT  PTR;
  DCL ITEM CHAR(WORDL)  BASED (TP);
   DCL   J  FB;
   DCL  TP  PTR;
      DCL  1 THING BASED (TP),
                  2 COUNT FB,
                  2 NAME PTR,
                  2 LEF PTR,
                  2 RIT PTR;
    TP=IT;
   TP=NAME;
   IF    I=0  THEN  PUT  LIST  ('$' || ITEM) ;
              ELSE  PUT  LIST  (ITEM );
   RETURN;
  END LISTI;
-
  LISTPRC: PROC (HEAD )  RECURSIVE;
  /*  PRINTS WHOLE TREE IN " POLISH " NOTATION . LUKASIEWICZ.  */
  /* DICTIONARY SORT  */
   DCL   HEAD  PTR;
    DCL  1 THING BASED (TP),
                  2 COUNT FIXED BIN,
              2 NAME PTR,
              2 LEF PTR,
              2 RIT PTR;
```

Program Listing (Continued)

```
      DCL  TP   PTR;
      IF     HEAD =  NULL THEN  RETURN;
     TP   =  HEAD ;
% IF DEBUG=0 % THEN % GO TO DB95;
        PUT SKIP EDIT(HEXD(TP),HEXD(NAME),HEXD(LEF),HEXD(RIT))
          (A,X(15),A,X(10),A,X(10),A,X(10));
% DB95:;
        PUT LIST(COUNT);
      CALL LISTI(TP,1B);
      IF LEF¬=NULL THEN CALL LISTPRC(LEF);
                   ELSE PUT SKIP ;
      IF RIT¬=NULL THEN CALL LISTPRC(RIT);
                   ELSE PUT SKIP ;
   RETURN;
 END LISTPRC;
-
 % IF DEBUG=0 % THEN % GO TO DB100;
 LISTRC:  PROC  (HEAD) RECURSIVE;
  /* SAME AS LISTPRC ONLY PRINTS CHARACTER STRING ITSELF  */
      DCL  HEAD PTR;
      DCL 1 THING BASED(TP),
                  2 COUNT FIXED BIN,
                  2 NAME PTR,
                  2 LEF PTR,
                  2 RIT PTR;
      DCL TP PTR;
      IF HEAD=NULL  THEN RETURN;
      TP=HEAD;
   CALL  LISTI(TP,0B);
      IF LEF¬=NULL THEN CALL LISTRC(LEF);
                   ELSE PUT LIST ('');
      IF RIT¬=NULL THEN CALL LISTRC(RIT);
                   ELSE PUT LIST ('');
      RETURN;
  END LISTRC;
 % DB100:;
-
-
-
  REORDER:
        PROC (HED,TP) RECURSIVE;
        DCL TP PTR;
        DCL HED PTR;
        DCL 1 THING BASED(TP),
              2 COUNT FIXED BIN,
              2 NAME PTR,
              2 LEF PTR,
              2 RIT PTR;
        IF TP = NULL THEN RETURN;
        IF LEF ¬= NULL THEN CALL REORDER(HED,LEF);
        CALL ADD(HED,TP);
        IF RIT ¬= NULL THEN CALL REORDER( HED,RIT);
        RETURN;
  END REORDER;
-
  % IF DEBUG=0 % THEN % GC TO DB200;
  TELL:  PROC (KIND,PT,MD);
  /*  FOR DEBUGGING  */
  DCL  ASD  PTR;
    DCL  MD  CHAR(1);
   DCL  MODE  CHAR (4);
```

Program Listing (Continued)

```
   DCL   KIND CHAR (*);
 DCL  PP  PTR;
 DCL  PRET  CHAR (8);
   DCL  PT  PTR;
   DCL 1 STUF BASED (PP), 2 ASF  PTR;
  DCL   VALU  CHAR (8);
  PRET =  HEXD(PT);
   MODE ='UNKN';
   IF     MD  =  'F'    THEN   MODE   =  'FREE'  ;
   IF     MD  =  'L'    THEN   MODE   =  'ALLC'  ;
   IF     MD  =  'P'    THEN  GO TO   POINT;
  PUT FILE(TALES) SKIP  LIST  (MODE,KIND,'AT  '||PRET);
 RETURN;
 POINT:
   MODE='FREE';
   PP=PT;
   ASD=ASF;
   VALU=HEXD(ASD);
  PUT FILE(TALES) SKIP  LIST  (MODE,KIND,'AT  '||PRET,'VALUE',VALU);
   RETURN  ;
 END TELL;
 % DB200:;

 END TREE;
FINIS:
        PUT PAGE;
        PUT SKIP LIST('NO OF DOCUMENTS PROCESSED=', NREQ-1);
  END TESTMT;
/*
//GO.ASPEN  DD DSNAME=CONCAL,UNIT=TAPE9,VOLUME=SER=ILR080,
//  DCB=(RECFM=FB,BLKSIZE=1615,LRECL=85),DISP=OLD,LABEL=(1,NL,,IN)
//GO.TALES  DD  SYSOUT=A,DCB=(LRECL=137,RECFM=VBA,BLKSIZE=3566),        X
//   SPACE=(TRK,40,RLSE)
//GO.PL1DUMP DD SYSOUT=A
//GO.SYSIN  DD  *
193,
  194,195,196,197,198,199,200,201,202,203,204,205,206,207,208,209,210,
  211,212,213,214,215,216,217,218,219,220,221,222,223,224,225,226,227,228,
229,230,231,232,233, 234,
999
  249,249,249,249,249,249,249,249,249,240,
999
  EMPLOYMENT OPPORTUNITIES
FOOD STAMPS
HOUSEHOLDS
INSTITUTIONS CODE
PUBLIC SOCIAL SERVICES
SNELLEN TEST
VISUAL ACUITY
WELFARE
  WORK CENTER
0
WITHIN
OF
THAT
ON
A
FIVE
WAS
IS
BY
```

Program Listing (Continued)

```
DURING
IN
NOT
OR
BOTH
BEING
FOR
THE
HAVE
IF
NO
THEY
ONE
WERE
TWO
AND
WHEN
WITH
WHICH
PRIOR
TO
AFTER
BECAUSE
0
/*
```

8. Sample Output

CAL_____SIAIS_____0027.0000

__A CAL. WELFARE AND INSTITUTIONS CODE, SEC. 19874

C ANY BLIND PERSON OF SUITABLE AGE, CHARACTER, AND QUALIFICATIONS, WHO IS A RESIDENT OF THE STATE OF CALIFORNIA, AND SUCH OTHER PHYSICALLY HANDICAPPED PERSONS SO QUALIFIED AS IT IS PRACTICABLE TO RECEIVE THEREIN, MAY BE ADMITTED FOR EMPLOYMENT IN ANY OPPORTUNITY CENTER UPON APPROVAL OF THE DEPARTMENT OF EDUCATION. THE RATE OF WAGES TO BE PAID WORKERS AND THE CONDITIONS OF EMPLOYMENT IN THE OPPORTUNITY WORK CENTERS FOR THE BLIND SHALL BE FIXED BY THE DEPARTMENT OF EDUCATION.

D WELFARE - PUBLIC SOCIAL SERVICES - RELIF LAW OF 1945

_SUBROUTINE SYMCON

INPUT CHARACTER=C DEC_VALUE= 3 BIN= 11000011 HEX_VALUE =C3 NEW_KAR= C

INPUT CHARACTER=A DEC_VALUE= 1 BIN= 11000001 HEX_VALUE =C1 NEW_KAR= A

INPUT CHARACTER=N DEC_VALUE= 21 BIN= 11010101 HEX_VALUE =D5 NEW_KAR= N

INPUT CHARACTER=Y DEC_VALUE= 40 BIN= 11101000 HEX_VALUE =E8 NEW_KAR= Y

INPUT CHARACTER=B DEC_VALUE= 2 BIN= 11000010 HEX_VALUE =C2 NEW_KAR= B

INPUT CHARACTER=L DEC_VALUE= 19 BIN= 11010011 HEX_VALUE =D3 NEW_KAR= L

INPUT CHARACTER=I DEC_VALUE= 9 BIN= 11001001 HEX_VALUE =C9 NEW_KAR= I

INPUT CHARACTER=N DEC_VALUE= 21 BIN= 11010101 HEX_VALUE =D5 NEW_KAR= N

INPUT CHARACTER=D DEC_VALUE= 4 BIN= 11000100 HEX_VALUE =C4 NEW_KAR= D

INPUT CHARACTER=P DEC_VALUE= 23 BIN= 11010111 HEX_VALUE =D7 NEW_KAR= P

INPUT CHARACTER=E DEC_VALUE= 5 BIN= 11000101 HEX_VALUE =C5 NEW_KAR= E

INPUT CHARACTER=R DEC_VALUE= 25 BIN= 11011001 HEX_VALUE =D9 NEW_KAR= R

INPUT CHARACTER=S DEC_VALUE= 34 BIN= 11100010 HEX_VALUE =E2 NEW_KAR= S

INPUT CHARACTER=O DEC_VALUE= 22 BIN= 11010110 HEX_VALUE =D6 NEW_KAR= O

Sample Output (Continued)

```
INPUT CHARACTER=N     DEC_VALUE= 21    BIN= 11010101    HEX_VALUE =D5    NEW_KAN= N

INPUT CHARACTER=O     DEC_VALUE= 22    BIN= 11010110    HEX_VALUE =D6    NEW_KAN= O

INPUT CHARACTER=F     DEC_VALUE=  6    BIN= 11000110    HEX_VALUE =C6    NEW_KAN= F

INPUT CHARACTER=S     DEC_VALUE= 34    BIN= 11100010    HEX_VALUE =E2    NEW_KAN= S

INPUT CHARACTER=U     DEC_VALUE= 36    BIN= 11100100    HEX_VALUE =E4    NEW_KAN= U

INPUT CHARACTER=I     DEC_VALUE=  9    BIN= 11001001    HEX_VALUE =C9    NEW_KAN= I

INPUT CHARACTER=T     DEC_VALUE= 35    BIN= 11100011    HEX_VALUE =E3    NEW_KAN= T

INPUT CHARACTER=A     DEC_VALUE=  1    BIN= 11000001    HEX_VALUE =C1    NEW_KAN= A

INPUT CHARACTER=B     DEC_VALUE=  2    BIN= 11000010    HEX_VALUE =C2    NEW_KAN= B
```

STATUTE C ANY BLIND PERSON OF SUITABLE AGE, CHARACTER, AND QUALIFICATIONS, WHO IS A RESIDENT OF THE STATE OF CALIF
ORNIA, AND SUCH OTHER PHYSICALLY HANDICAPPED PERSONS SO QUALIFIED AS IT IS PRACTICABLE TO RECEIVE THEREIN, MAY BE ADMITTED FOR EMPLO
YMENT IN ANY OPPORTUNITY CENTER UPON APPROVAL OF THE DEPARTMENT OF EDUCATION. THE RATE OF WAGES TO BE PAID WORKERS AND THE CONDITIO
NS OF EMPLOYMENT IN THE OPPORTUNITY WORK CENTERS FOR THE BLIND SHALL BE FIXED BY THE DEPARTMENT OF EDUCATION. D WELFARE - PUBLIC S
OCIAL SERVICES - RELIEF LAW OF 9099

SUBROUTINE HEXCON

```
SS4040C340C1D5E840C2D3C9D5C440D7C5D9E2D6D540D6C640E2C4C9E3C1C2D3C540C1C7C5D5E840C3C9C1D9C1C3E3C5D9E340C1D5C440D8E4C1D3C9C6C9C3C1E3C9D6
D5E26B40E6C8D640C9E240C140D9C5E2C9C4C5D5E340D6C640E3C8C540E2E3C1E3C540D6C640C3C1D3C9C6D6D9D5C9C1E340C1D5C440E2E4C3C840D6E3C8C5D940D7
C8E8E2C9C3C1D3D3E840C8C1D5C4C9C3C1D7D7C5C440D7C5D9E2D6D5E240E2D640D8E4C1D3C9C6C9C5C440C1E240C9E340C9E240D7D9C1C3E3C9C3C1C2D3C540E3D6
40D9C5C3C5C9E5C540E3C8C5D9C5C9D5684004C1E340C2C540C1C404C9E3E3C5C440C6D6D340C5C4D7D3D6E8D4C5D5E340C9D540C1D5E840D6D7D7D6D9E3E4D5C9E3
E840C3C5D5E3C5D940E4D7D6D540C1D7D7D9D6E5C1D340D6C640E3C8C540C4C5D7C1D9E3D4
```

SUBROUTINE LOOKUP_____INDEX ITEMS IN CONTEXT

EMPLOYMENT OPPORTUNITIES
SEARCH UNSUCCESSFUL

FOOD STAMPS
SEARCH UNSUCCESSFUL

HOUSEHOLDS
SEARCH UNSUCCESSFUL

INSTITUTIONS CODE
SEARCH UNSUCCESSFUL

Sample Output (Continued)

```
PUBLIC SOCIAL SERVICES
XED BY THE DEPARTMENT OF EDUCATION.    D WELFARE - PUBLIC SOCIAL SERVICES - RELIF LAW OF 9099

SNELLEN TEST
SEARCH UNSUCCESSFUL

VISUAL ACUITY
SEARCH UNSUCCESSFUL

WELFARE
HALL BE FIXED BY THE DEPARTMENT OF EDUCATION.    D WELFARE - PUBLIC SOCIAL SERVICES - RELIF LAW OF 9099

WORK CENTER
D THE CONDITIONS OF EMPLOYMENT IN THE OPPORTUNITY WORK CENTERS FOR THE BLIND SHALL BE FIXED BY THE DEPARTMENT O
```

Sample Output (Continued)

```
LISTPRC(HEAD)
  DICTIONARY SORT
        1                ADMITTED
        1                AGE
        2                ANY
        1                APPROVAL
        1                AS
        3                BE
        2                BLIND
        1                C
        1                CALIFORNIA
        1                CENTER
        1                CENTERS
        1                CHARACTER
        1                CONDITIONS
        1                D
        2                DEPARTMENT
        2                EDUCATION
        2                EMPLOYMENT
        1                FIXED
        1                HANDICAPPED
        1                IT
        1                LAW
        1                MAY
        2                OPPORTUNITY
        1                OTHER
        1                PAID
        1                PERSON
        1                PERSONS
        1                PHYSICALLY
        1                PRACTICABLE
        1                PUBLIC
        1                QUALIFICATIONS
        1                QUALIFIED
        1                RATE
        1                RECEIVE
        1                RELIF
        1                RESIDENT
        1                SERVICES
        1                SHALL
        1                SO
        1                SOCIAL
        1                STATE
        1                SUCH
        1                SUITABLE
        1                THEREIN
        1                UPON
        1                WAGES
        1                WELFARE
        1                WHO
        1                WORK
        1                WORKERS
        1                9099
```

Sample Output (Continued)

```
LISTPRC(HED)
 FREQUENCY LIST
         3                BE
         2                ANY
         2                BLIND
         2                DEPARTMENT
         2                EDUCATION
         2                EMPLOYMENT
         2                OPPORTUNITY
         1                ADMITTED
         1                AGE
         1                APPROVAL
         1                AS
         1                C
         1                CALIFORNIA
         1                CENTER
         1                CENTERS
         1                CHARACTER
         1                CONDITIONS
         1                D
         1                FIXED
         1                HANDICAPPED
         1                IT
         1                LAW
         1                MAY
         1                OTHER
         1                PAID
         1                PERSON
         1                PERSONS
         1                PHYSICALLY
         1                PRACTICABLE
         1                PUBLIC
         1                QUALIFICATIONS
         1                QUALIFIED
         1                RATE
         1                RECEIVE
         1                RELIF
         1                RESIDENT
         1                SERVICES
         1                SHALL
         1                SO
         1                SOCIAL
         1                STATE
         1                SUCH
         1                SUITABLE
         1                THEREIN
         1                UPON
         1                WAGES
         1                WELFARE
         1                WHO
         1                WORK
         1                WORKERS
         1                9099
```

IV. PROGRAM INDX AND PROGRAM CONCORD

1. Preparation of Texts for Input

1.1 Textual Input Record Format

Input to both programs is via eight-character input records. The input record format is described below by input positions.[1]

Positions 1 to 71 - Input record positions 1 to 71 are used to contain free format textual data in the form of words and/or punctuation marks. Each word is delineated by blanks or punctuation marks.

Position 72 - Position 72 is blank or contains one of the following special characters:

1. Dash (-) signals a continuation of a word between succeeding input records. If a complete word does not end in position 71, enter the dash in position 72, and continue the word beginning in position 1 of the following record. The word counter will be incremented only once.

2. Commercial "at" sign (@) signals a continuation of a sentence between succeeding records. This operator is valid only for type processing POET or MILT. All other types of processing will cause the operator to be ignored. The placing of this operator in position 72 causes sentence counters not to be incremented. This option is necessary because a line of poetry (either in the form indicated by MILT--which may be divided into books, or cantos, or

---

[1]Note that program CONCORD in its present form can only cope with chapters and paragraphs, but that it is only a matter of a little time and effort to incorporate all the options allowed for in Program INDX.

something analogous, as well as lines) may well run over onto a second printed line even though the initial line and the run-over comprise just one poetic line.

3. Pound sign (#) signals the continuation of both the word and sentence between input records. Use of this operator will cause the word counter not to be reset and the sentence counter will not be incremented. This operator is valid only for type processing POET or MILT.

Positions 73 and 74 - If type processing is MILT the program expects to find a volume number in two-digit format in positions 73 and 74. The index for Paradise Lost, the work for which the MILT option is currently being used, gives the book number in columns 73 and 74 and the line number in columns 75 - 78. For example, 110232 would refer to Book 11, line 232. The line number may be right-justified in columns 75 - 80, if the programmer prefers.

Positions 75 - 80 - These input record positions are for use in sequencing of the input records. Any combination of character/digits may be used providing they form a logical collating sequence. The collating sequence is not program checked. Under type of processing, OPLA (out-of-Sequence Play: for a play such as the Ginn & Company, Kettredge edition of Hamlet, in which the printed line numbers and the number of lines referenced on the page seem not to correspond) positions 1 - 68 contain free format input data, positions 69 - 72 contain a play line number which is equated to a sentence number, and positions 73 - 80 may be used for sequencing.

1.2 Program Control Key Words

The INDX program recognizes two types of key words; 1) processing key words, those key words specifying how data is to be processed by the program and, 2) textual key words which are used with passages which have been excerpted from longer texts (such as sections of Hume's History of England) to indicate the location of the excerpt within the text as a whole.

1.2.1 Processing Key Words

Processing key words are passed to the program through a Job Control Language (JCL) parameter character string or the first input record of the input data set. Position notation is used, i.e., the program expects to find key words at specified locations within the character string. Each processing key word will be discussed according to its position.

Positions 1 - 4 - Type of Processing - These positions contain a code indicating which type of processing is to be performed. Seven type processing key words are allowed. Entry of other than one of the following key words will cause the program to terminate. The processing key words recognized are:

FRST - This key word must be passed to the INDX program through the JCL parameter field when all key words are to be found on the first record of the input data set.

PROS - This key word should be used for normal textual data. The volume, chapter, paragraph and sentence indexing counters are reset and incremented by textual key words contained in the text. Indexed words and/or punctuation are written

as they are distinguished on the input records. The word continuation symbol is recognized (input record position 72) but the line and word/line continuation symbols are used just with POET and MILT.

PLAY - This key word causes processing in the same manner as PROS, but with the addition of the processing key word, 'STAGE DIRECTION'. Stage directions are processed as though normal text (refer to 1.2.2 for STAGE DIRECTION key word format). When stage directions are processed, they appear continuously in the output because each stage direction is preceded by an asterisk. This asterisk remains affixed as the first character of the word.

SPOK - This key word causes processing in the same manner as PROS. However, different textual key words are recognized for resetting counters. (refer to Section 1.2.2)

POET - This key word causes printing of the complete input record rather than individual words and/or punctuation as they are recognized in the input record. A check is not made for textual key words. The inclusion of textual key words in the input text will cause them to be indexed in a normal manner. Volume, chapter and paragraph index counters are each to the value one. The sentence counter is incremented each time an input record is read. The word counter is reset each time the sentence counter is incremented. Word, sentence,

and word/sentence continuation symbols in input record positions 72 are recognized.

MILT - MILT is a special case of POET. General processing is the same except the volume indexing counter is set via data in positions 73 and 74 of the input record.

OPLA - The use of OPLA causes the program to print-out the entire record as with POET. However, the sentence indexing counter is reset from input record positions 69 - 72 and the PLAY textual key words are recognized including stage directions when the processing key word STAGE DIRECTION has been specified.

Positions 6 - 10 - print control. These positions follow a comma, which occurs after the processing key word and indicate whether or not the indexed words or punctuation marks are to be printed.

PRINT - indicates that textual data is to be printed.

NOPRT - indicates that textual data is not to be printed.

If these positions are blank, no printing is assumed. Program control information such as index counter reset values and ending summary messages are printed independent of the print control key word.

Positions 12 - 26 - stage directions. These positions follow a comma after the print control key word. Specifying the key word STAGE DIRECTION will cause the processing of stage directions (and/or punctuation) as though they were textual words. When the processing key word is PLAY or OPLA the omission of this

key word will cause stage directions to be ignored.

Examples for use of processing key words:

1) To process Prose, to print textual data as processed, and to introduce the key words through the JCL parameter field, the OS/360 JCL EXEC card would be:

```
//    EXEC XXXX,PARM.GO='PROS,PRINT'
```

(where 'XXXX' indicates catalogued procedure to be executed.)

utilizing the first input record to contain processing key words, the JCL EXEC card would be:

```
//    EXEC XXXX,PARM.GO='FRST'
```

and the first input record would contain, beginning in column 1

```
PROS,PRINT.
```

2) To process a play, not printing textual data, but processing stage directions, information furnished through JCL would be

```
//     EXEC XXXX,PARM.GO='PLAY,NOPRT,STAGEDIRECTION'
```

1.2.2 Textual Key Words

Textual key words are of two types:

1) those key words used to reset indexing counters
2) those textual key words used to increment indexing counters.

Reset Key Words

Key words used to reset indexing counters are prefaced by a dollar sign, followed by the counter name that is to be

reset, followed by a value that the counter is to be reset to. Only one key word may appear on an input record. Input positions after the reset value are ignored and not processed. If processing key word is PROS, four reset key words are allowed:

$VOLUME followed by a decimal integer, XX, will reset the volume to that value.

$CHAPTER XXX will reset the chapter indexing counter to any value up through 3 decimal digits.

$PARAGRAPH XXX will reset the paragraph indexing counter to any value up through 3 decimal digits.

$SENTENCE XXXX will reset the sentence indexing counter to any value up through 5 decimal digits.

For processing key word PLAY or OPLA, two reset key words are allowed:

$ACT XXX will reset the chapter indexing counter to any value up through 3 decimal digits.

$SCENE XXX will reset the paragraph indexing counter to any value up through 3 decimal digits.

The volume indexing counter will equal the value one.

If the processing key words is SPOK, for spoken word, three reset key words are allowed:

$SERIES XX will reset the volume indexing counter to any value up through two decimal digits.

$SESSION XXX will reset the chapter indexing counter to any value up through three decimal digits.

$SPEAKER XXX will reset the paragraph indexing counter to any value up through three decimal digits.

For processing key words other than the four specified above, the textual key words to reset indexing counters are ignored. If textual key words appear in the text they will be treated as normal textual data and be indexed.

Increment Key Words

If the processing key word is PROS, the normal end-of-sentence punctuation, e.g., period, question mark, quotation mark, will increment the sentence indexing counter. When the sentence indexing counter is incremented, the word indexing counter is reset to one. If the processing key word is PROS, PLAY or SPOK a double period will increment the paragraph counter by one, reset the sentence indexing counter to one, and reset the word indexing counter to one. Three dollar signs in a row within the text will increment the chapter indexing counter by one, reset the paragraph indexing counter to one, sentence indexing counter and the word indexing counter to one. Similarly, four dollar signs in a row will increment the volume indexing counter by one, reset chapter, paragraph, sentence, and word indexing counters to one. In all cases the double period, three dollar signs and four dollar signs notations are not outputted as text as is the case in Sedelow's version.

2. Program INDX

2.1 Identification

Title	INDX
User Group	Institute of Library Research
Category	CIS Text Processing
Source Language	PL/1
Machine Configuration	IBM 360/91, a tape, printer, a disk
Space Requirements	Step Region:
	PL1L 162K
	GO 150K
	Core Used 98K
Author of original version	Sally Sedelow
Adapted to 360/91 by	G. Silva
Date	March 1970

2.2 Purpose

To read input text from cards in stream form.

To produce a word index from the text.[1]

To produce a word frequency list from the text.[2]

2.3 Output Record Format

Output from the indexing phase is fixed length, 46 character records. Each record contains one word and/or punctuation mark and its associated indexing information. Record format is as follows:

---

[1] A word index is an alphabetically ordered list of all items in a text with complete numerical reference to each passage where the term occurs, plus a frequency count for each item.

[2] A word frequency list is simply the list of all items in a text arranged in decreasing order of frequency.

Positions 1 - 4 - Four-byte variable count field. This field contains in binary format, the length of the entire variable length record.

Positions 5 and 6 - Two-byte length of the indexed word

Positions 7 - 9 - Two-byte volume number

Positions 10 - 12 - Three-byte chapter number

Positions 13 - 15 - Three-byte paragraph number

Positions 16 - 20 - Five-byte sentence number

Positions 21 - 25 - Two-byte word number within sentence

Positions 26 - 46 - Word

Output of records is accomplished by using the PL/1 PUT STRING function which converts the indexing counters from internal fixed binary format to character-string format.

Note

For purposes of testing the program, the output format was simplified as follows:

Positions 1 - 20 - Word

Positions 21 - 24 - Four-byte chapter number

Positions 25 - 28 - Four-byte paragraph number

Output from the dictionary sort (Word Index) takes the form illustrated in subsection 2.8.1, and output from the frequency sort phase (Word Frequency List) is illustrated in subsection 2.8.2.

2.4 Checkout

The program was tested on Oscar Wilde's "The Canterville Ghost", a short story with a total of 13328 words and punctuation marks. These were indexed, sorted into dictionary order, and resorted into frequency order.

2.4.1 Timing

The test was run in two steps, the indexing phase and the sorting phase.

Indexing phase took	43.25s cpu time
The two sorts took	38.49s cpu time
Total cpu time	81.74s

i.e., approximately 7 ms /1000 input items

2.4.2 Error Checking

None

2.4.3 Cost

The cost for the whole operation was:

Indexing phase	$13.94
Sort, word index	$15.17
Sort, frequency list	
Total	$29.11

i.e., approximately $2.18 / 1000 input items

2.4.4 Remarks

Sedelow's program is written in PL/1 and was developed on the IBM/360 Model 75. PL/1 is a problem-oriented language which is designed to be machine independent, and therefore the program was expected to run on any machine configuration which has a PL/1 compiler and a random-access memory extension (such as a disk).

Sedelow's program did indeed run on the IBM/360 Model 91. A few changes were necessary, however, some of minor significance, such as redeclaring fixed variables binary fixed, others more important.

A section was added in lieu of subroutine PUTOUT, and another for subroutine CLOSEOUT, since these were not included in the listing published in the report.

One slightly more significant change was the addition of a section in subroutine FORM, which inserts a blank before and after all punctuation marks in the text except when a period follows a period (signal for end of paragraph). It was felt that the requirement of Sedlow's program that words and punctuation marks in the text be delimited by blanks was too rigid, and that this could become an unwanted source of error introduced at the keypunching stage.

Sedelow's program looks very much like a statement for statement translation from a FORTRAN-like language. For example, in subroutine FORM words are built up character by character. This is a somewhat tedious coding method to adopt, considering that PL/1 offers such shortcuts as the INDEX, SUBSTR and LENGTH functions. Had time permitted, I would have made far more extensive changes than the little remedial additions mentioned above.

The original version was so far as possible left intact. The statements replaced by alternative code (my own) were in most cases turned into comments by the use of /*. . . .*/.

2.5 Program Description

Main Steps

The index program is designed to cope with texts in prose, poetry, and dramatic forms, as well as with transcriptions of

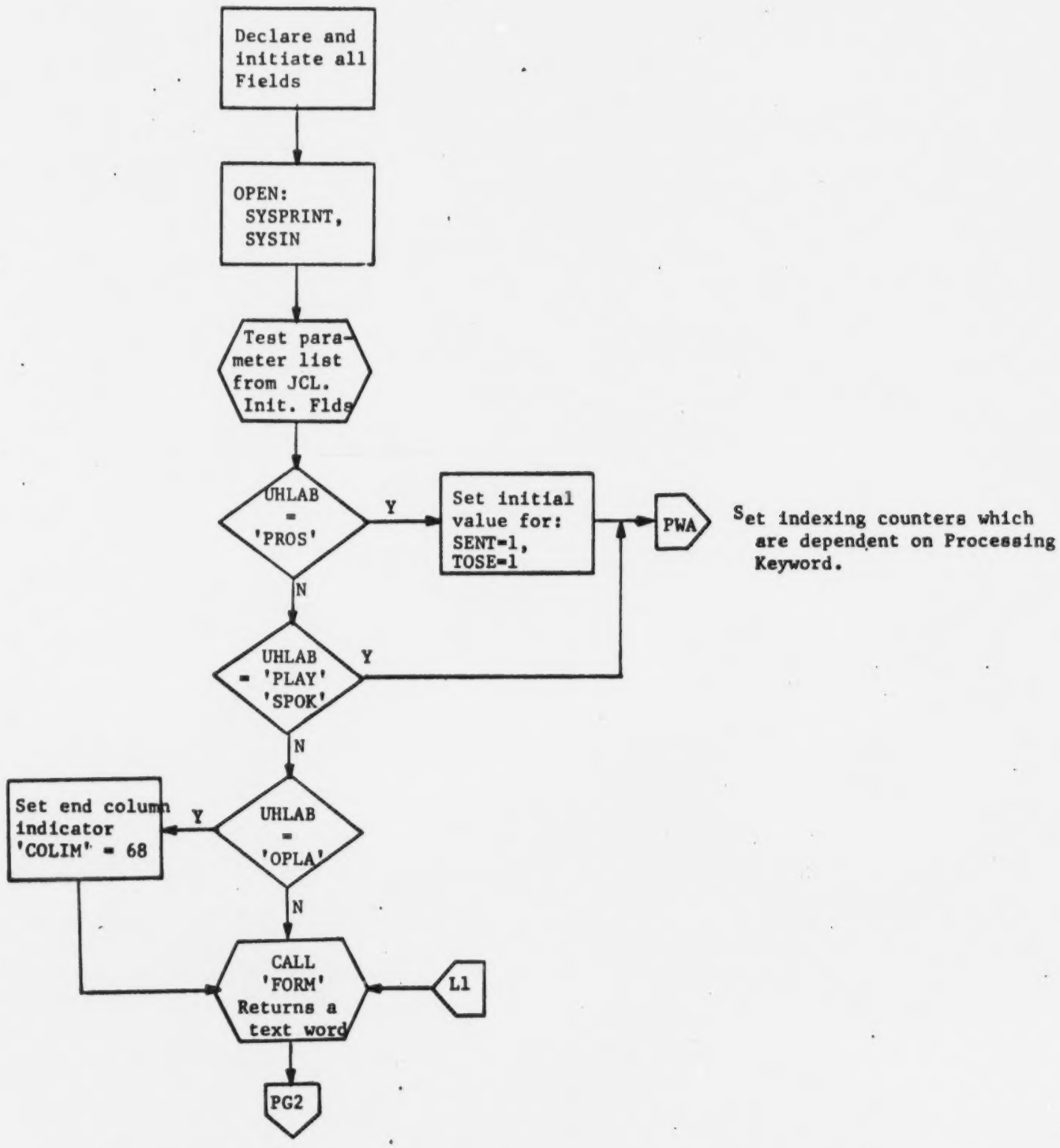
oral discourse. The program reads input text from cards and produces records each containing one textual word or punctuation mark, and index information showing where the word/punctuation mark occurred in the text. Each word/punctuation mark is indexed as to volume, chapter, paragraph, sentence, and word within sentence, and word within sentence location. These records are stored in a temporary file which forms the input to Program GAMMA (see section IV.4.1)

Program GAMMA produces the Word Index and the Word Frequency List. Program INDX keeps count of the number of volumes, chapters, paragraphs, words and punctuation marks processed, and prints them when the end of the text is reached.

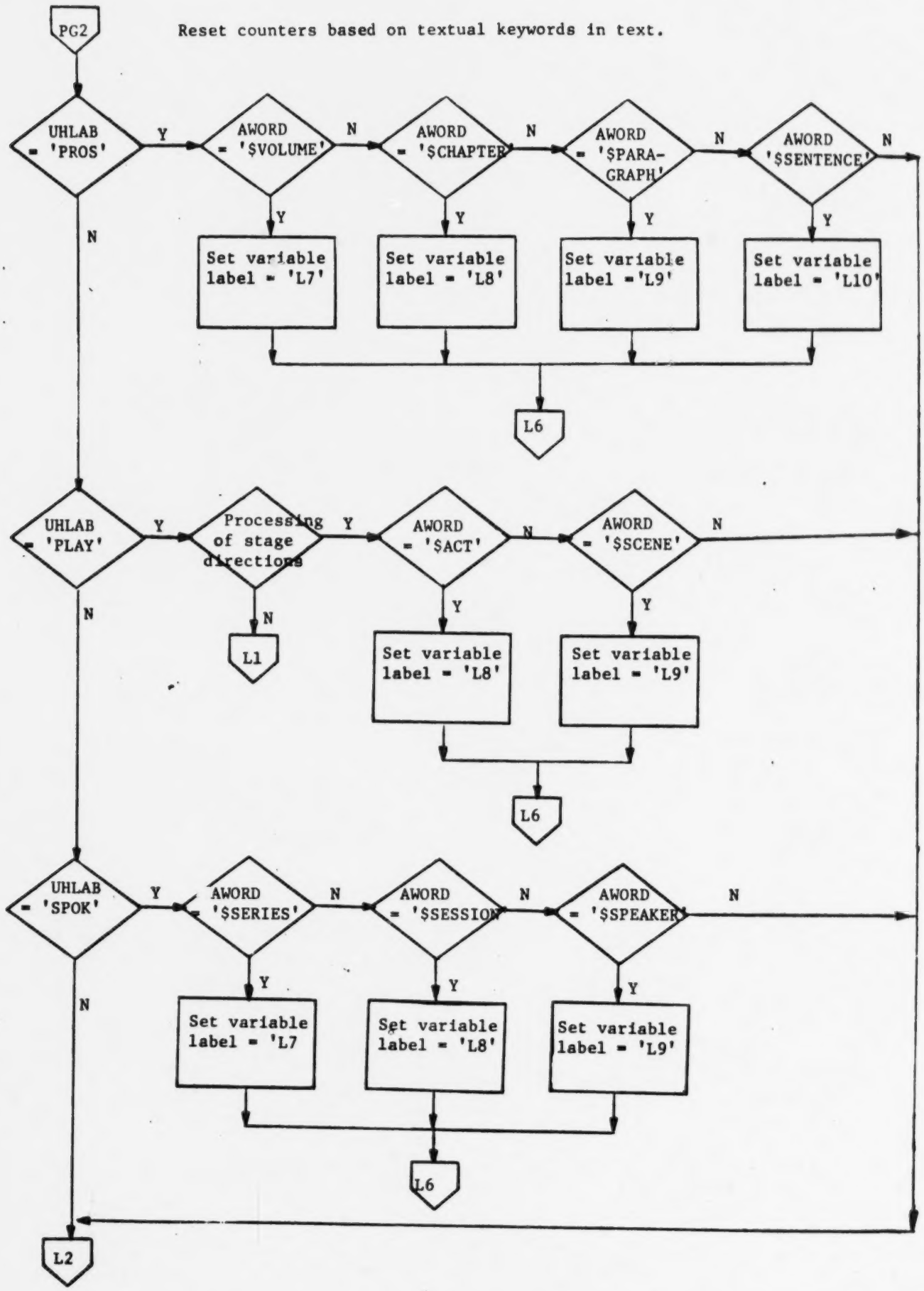
2.6 Flowchart for indexing phase

The flowchart appended is the one published in Sedelow's report. The changes made in my version have not been marked. However, the program listing in the next section is a listing of the current version, and gives an indication of the whole deck set-up including the JCL and some data.

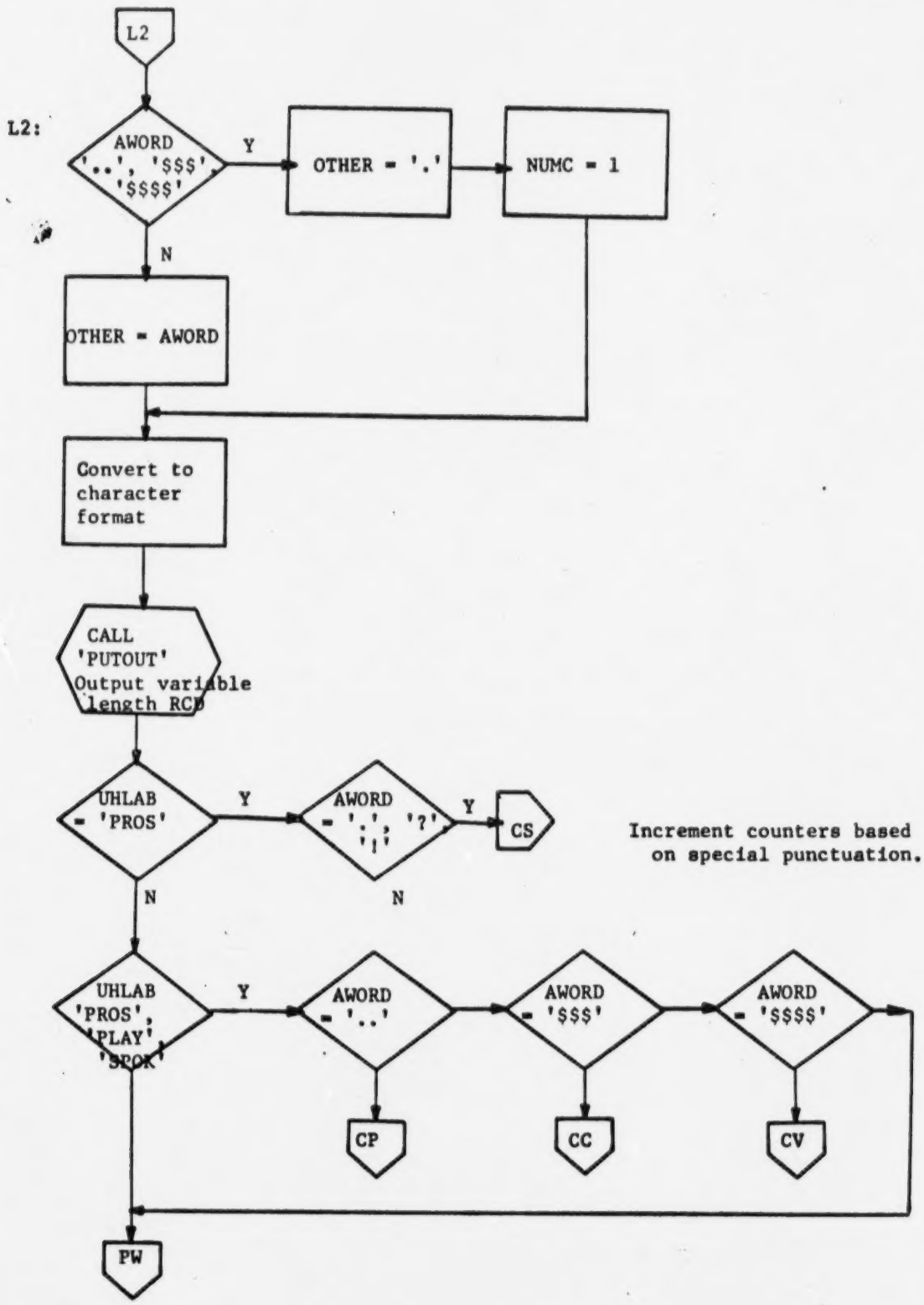
2.6 Flow Chart



INDEX Program: Main Routine

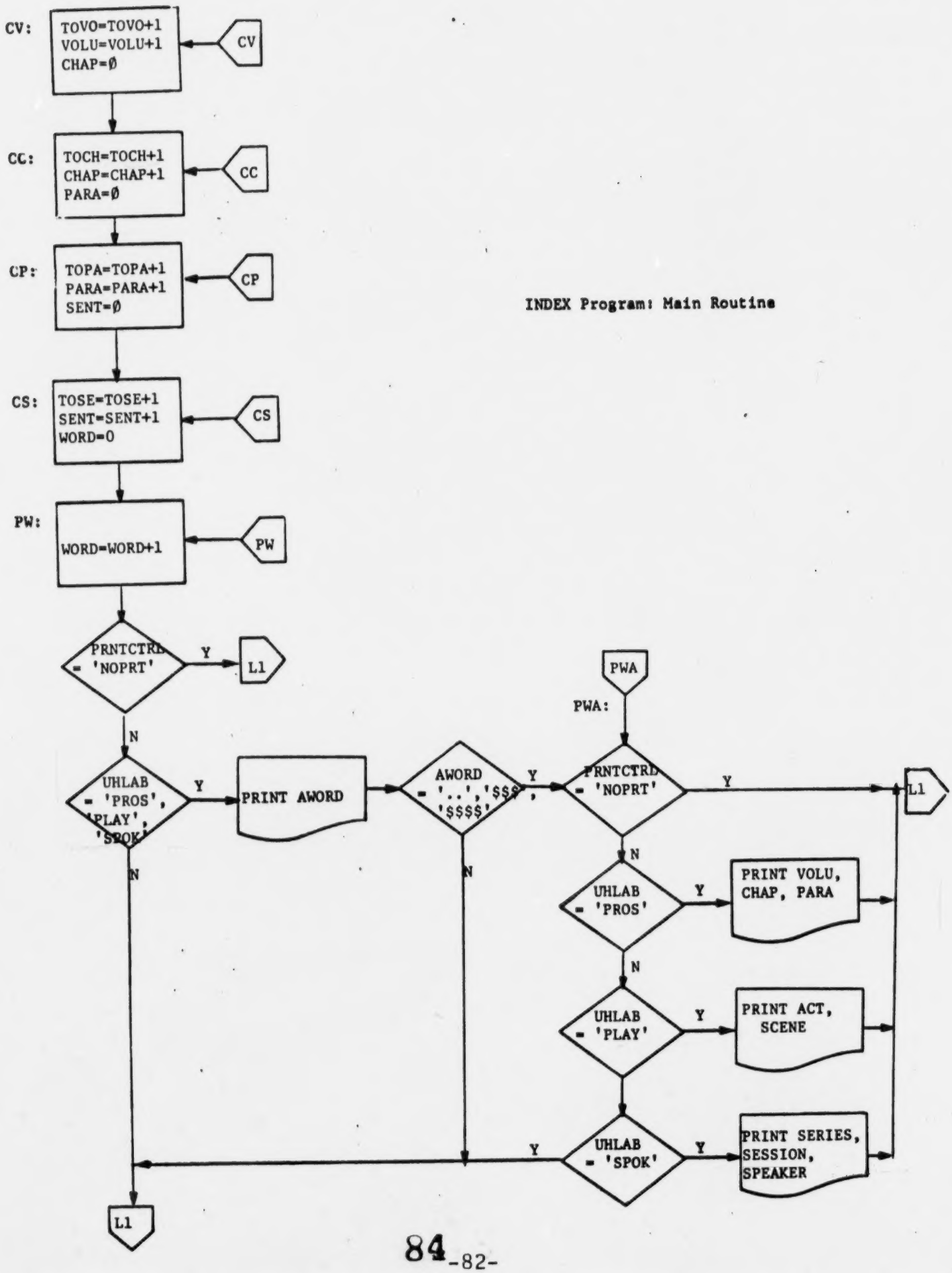


INDEX Program: Main Routine

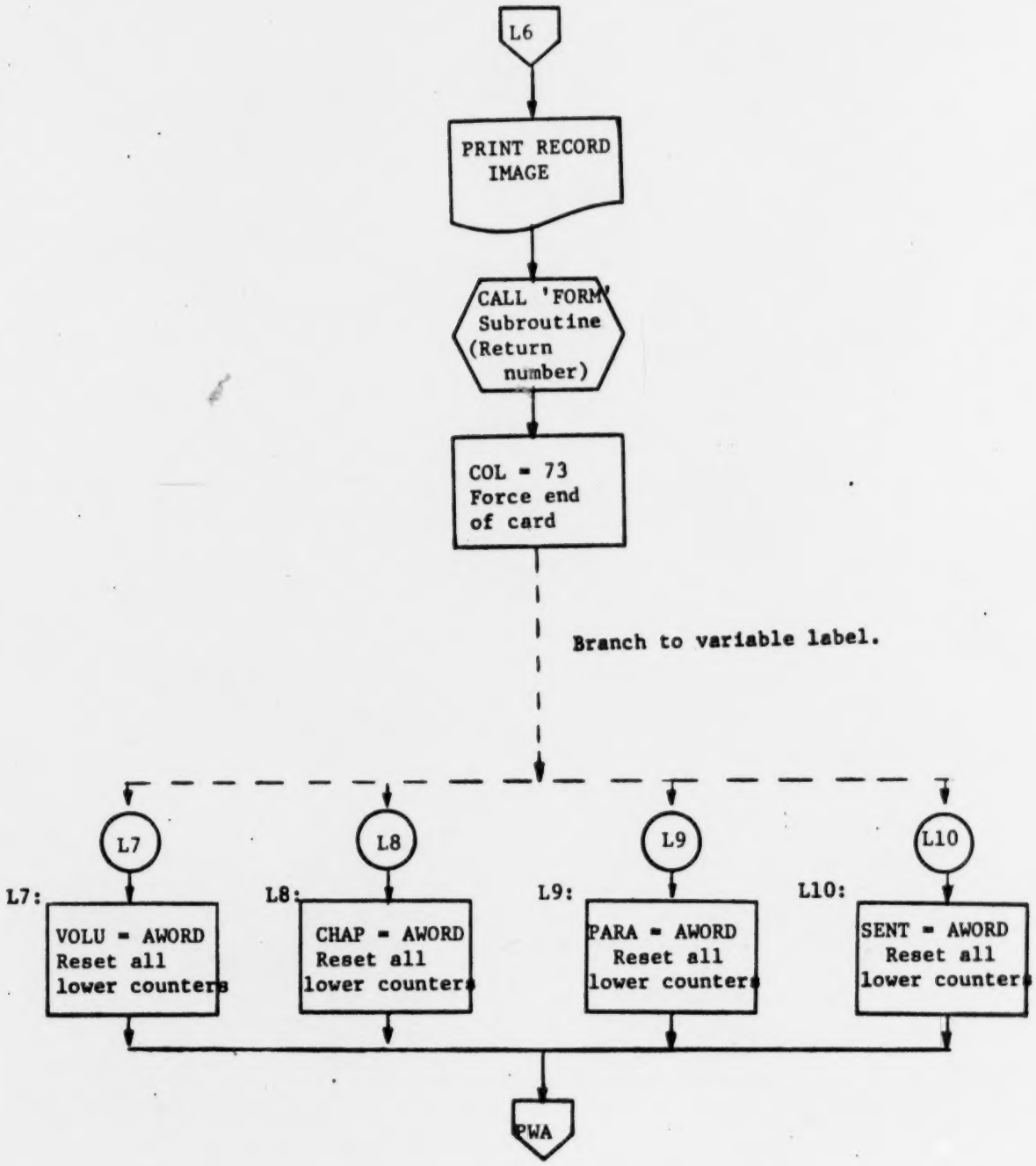


Increment counters based on special punctuation.

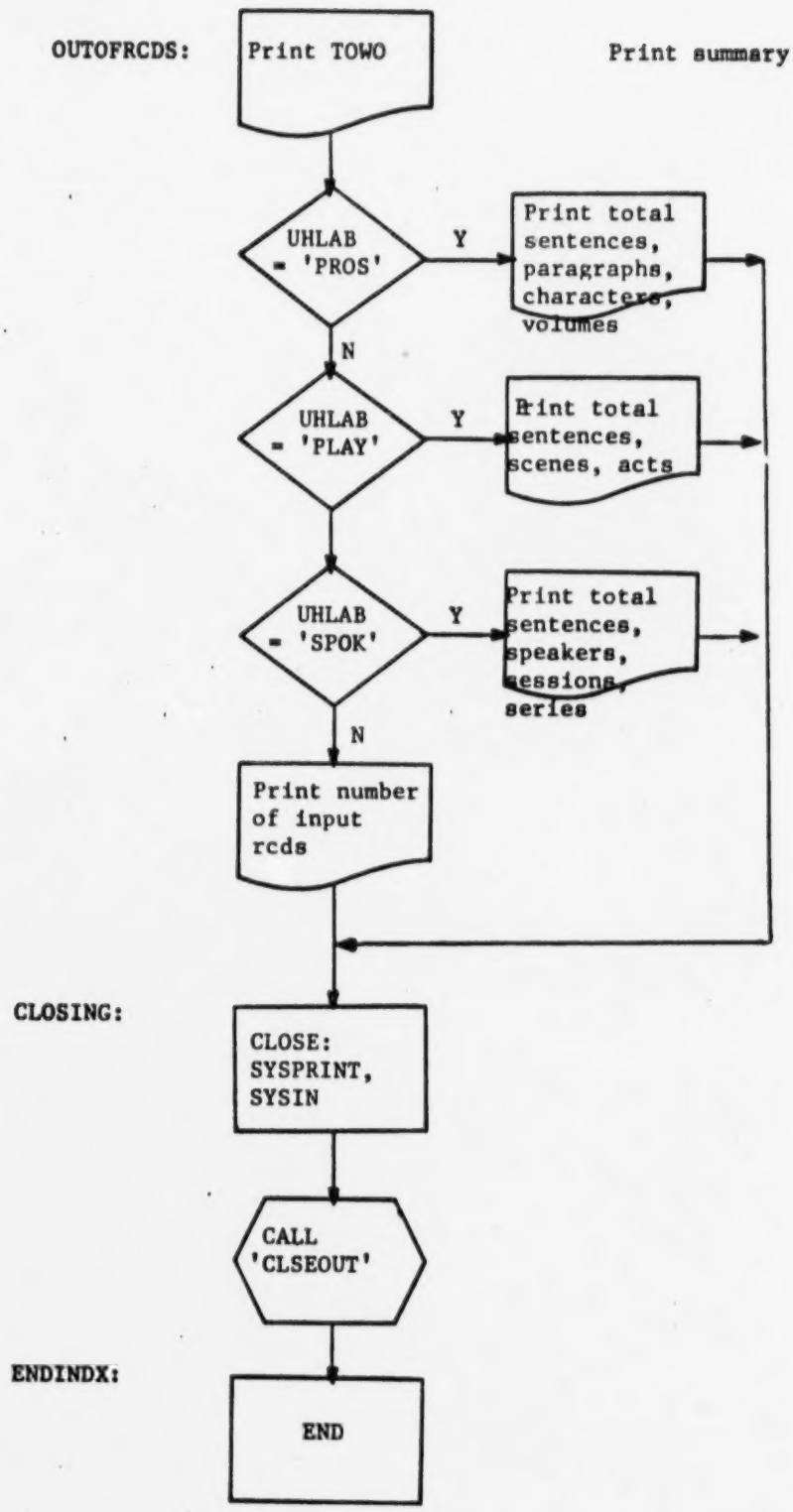
INDEX Program: Main Routine



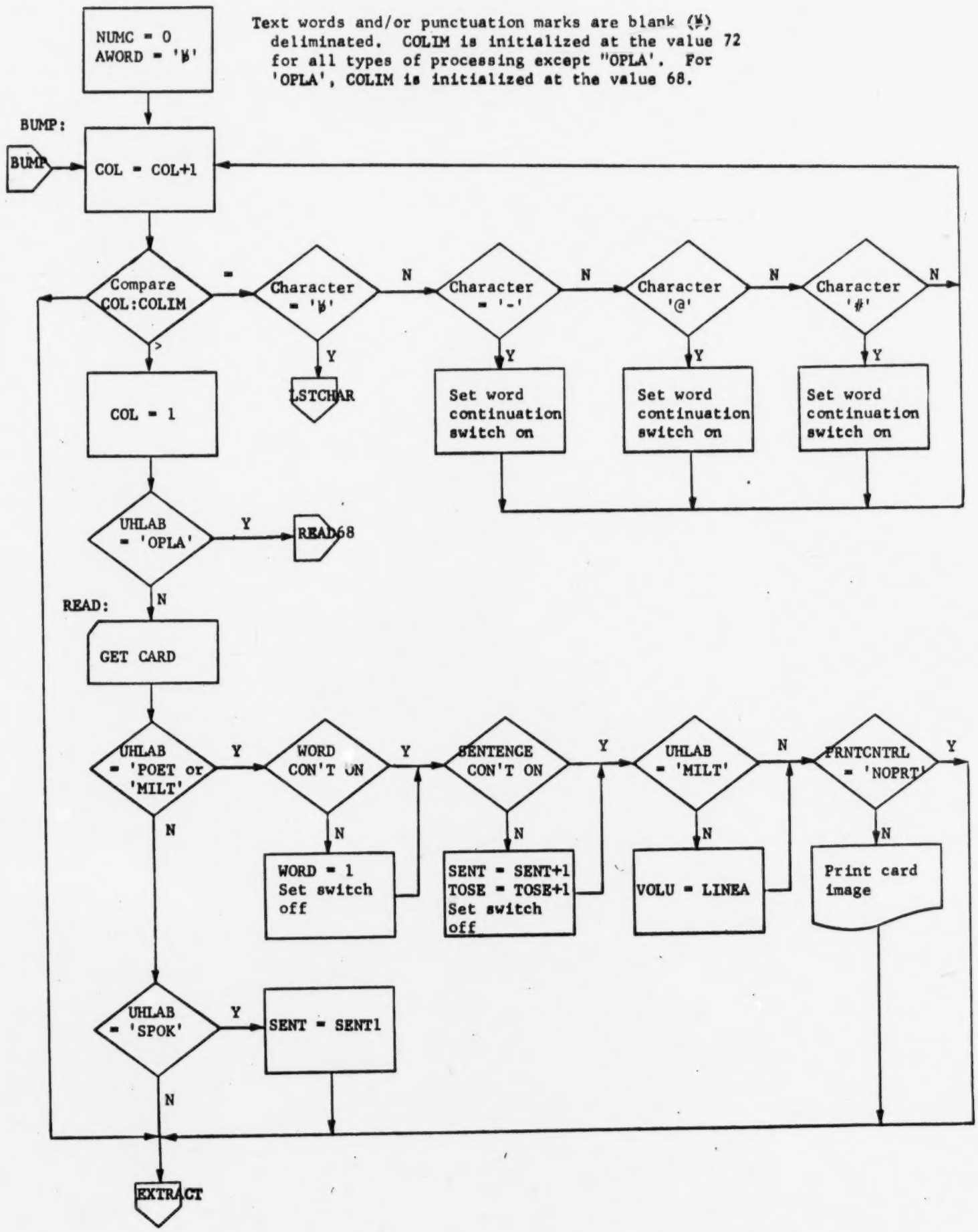
INDEX Program: Main Routine



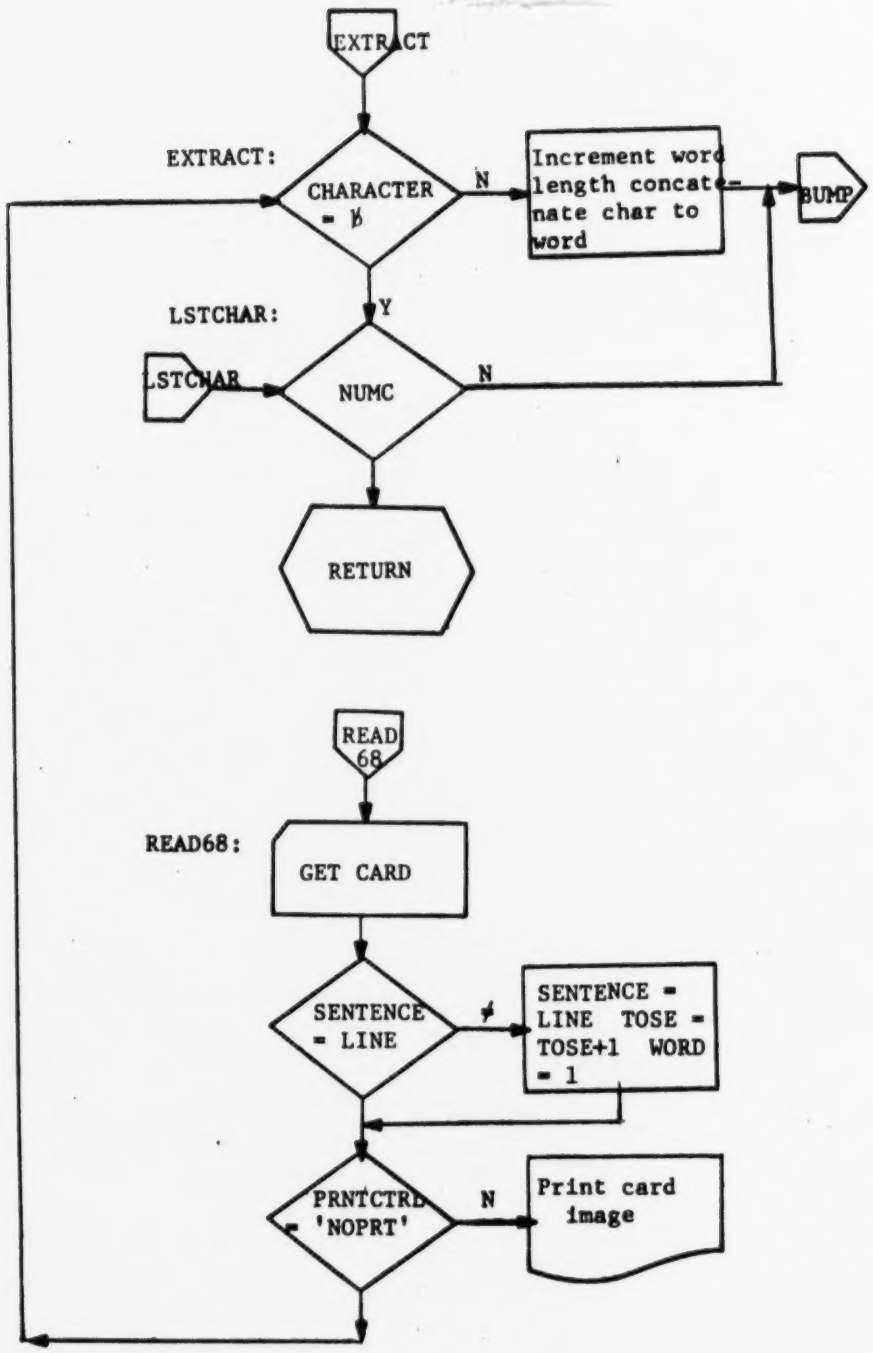
INDEX Program: Main Routine



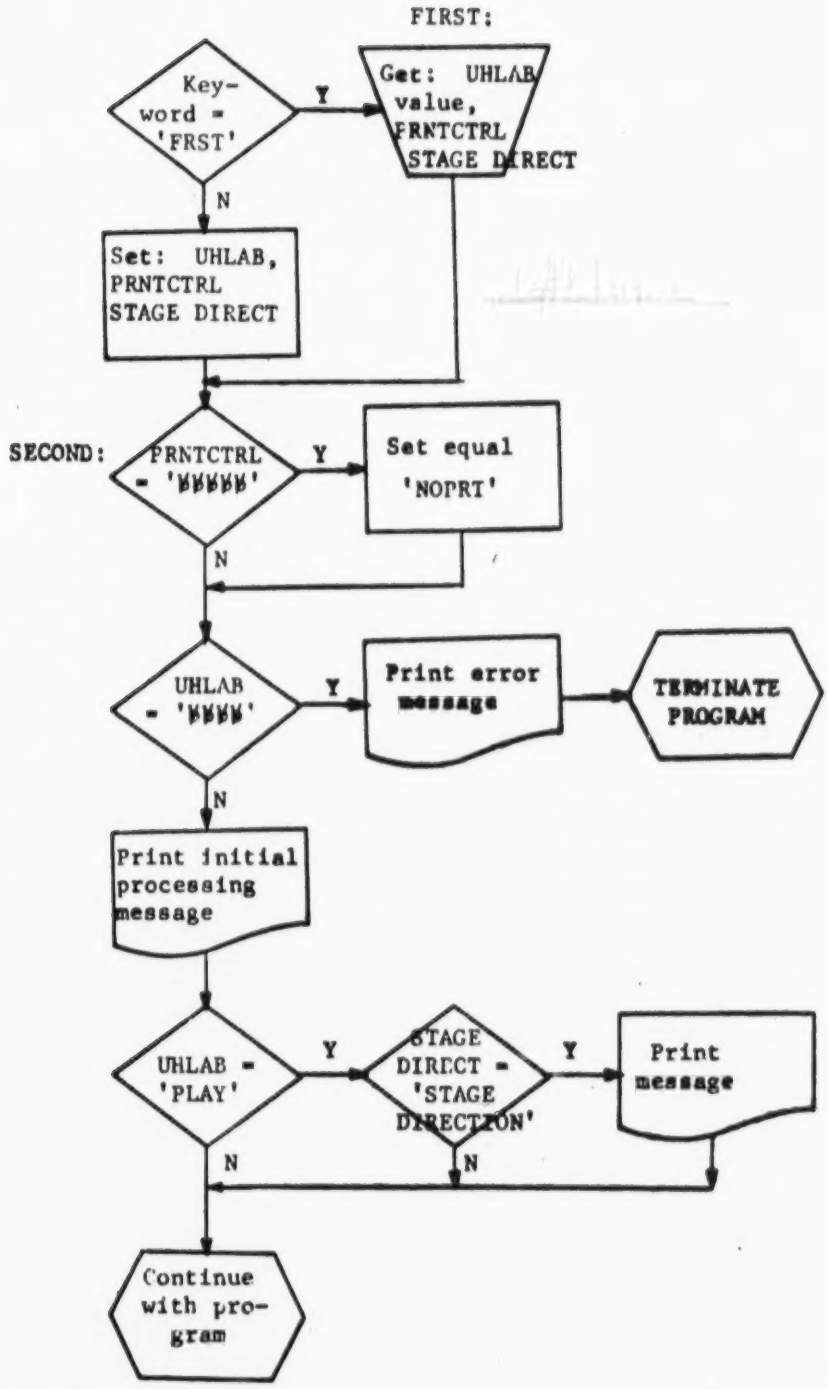
INDEX Program: Main Routine



INDEX Program: Subroutine FORM



INDEX Program: Subroutine FORM



TESTING PARAMETER LIST
Passed through JCL EXEC statement.

2.7 Program Listing

```
//NRB4C9GS  JOB  '59,1C000,R',SILVA
// EXEC PL1LFCG,PARM.GO='FRST',PG='(5C,20)'
//PL1L.SYSLIN  CC  SPACE=(TRK,(20,10))
//PL1L.SYSUT1  CC  SPACE=(TRK,(20,10))
//PL1L.SYSUT3  CC  SPACE=(TRK,(20,10))
//SYSIN DD *
  (SUBSCRIPTRANGE):(STRINGRANGE):
    INDX: PROC (PARM) OPTIONS(MAIN);
 /*  THE INDEX PROGRAM INDEXES TEXTUAL WORDS AND/OR PUNCTUATION MARKS */
 /*  PUNCHED ON DATA PROCESSING CARDS.  WORDS AND/OR PUNCTUATION MUST */
 /*  BE DELIMITED BY BLANKS.    THE PROGRAM INDEXES THE TEXT WORDS,   */
 /*  ETC. BY VOLUME NUMBER, CHAPTER NUMBER, PARAGRAPH NUMBER,         */
 /*  SENTENCE NUMBER, AND WORD NUMBER WITHIN SENTENCE.                */
 /* PUTOUT AND CLOSEOUT ARE INTERNAL SUBROUTINES WHICH DO NOT EXIST IN
        THIS DECK */
      DECLARE PARM CHAR (40) VARYING;  /* DATA FROM JCL 'EXEC CARD */
        DCL CARDIMAGE(90) CHAR(1),
            TEMP   CHAR (40) INITIAL (' '),
            UHLAB CHAR (4) INITIAL (' '),/* TYPE OF PROCESSING  */
            PRNTCTRL CHAR (5) INITIAL (' '),  /* PRINT, NOPRT PARM  */
            STAGEDIRECT CHAR (15) INITIAL (' '),
            TOVO FIXED BIN INITIAL (1),  /* TOTAL VOLUMES          */
            TOCH FIXED BIN INITIAL (1),  /* TOTAL CHAPTERS         */
            TOPA FIXED BIN INITIAL (1),  /* TOTAL PARAGRAPHS       */
            TOSE FIXED BIN INITIAL (0),  /* TOTAL SENTENCES        */
            TOWO FIXED BIN INITIAL (0),  /* TOTAL WORDS            */
            L LABEL (L7,L8,L9,L10) ;
      DECLARE  /* USED IN SUBROUTINE 'FORM'.                        */
            SEQUENCE CHAR (8) INITIAL (' '),
            LINE     FIXED BIN INITIAL (0),
         LINEA CHAR(2) INITIAL(' '),
            COL      FIXED BIN INITIAL (73),
            COLIM FIXED BIN INITIAL (72),  /* NUMBER OF COLUMNS   */
            WRDCONT FIXED BIN     INITIAL (0),
            LINECONT FIXED BIN     INITIAL (0);
                                   /* TO BE PROCESSED     */
      DECLARE    /* USED IN OUTPUT RECORD.                          */
            CNST FIXED BIN     INITIAL (18),  /* LENGTH OF FIXED
                                   PORTION OF RECORD     */
            NUMC FIXED BIN INITIAL (0),  /* NUM OF CHAR IN WORD   */
            VOLU FIXED BIN INITIAL (1),  /* VOLUME NUMBER         */
            CHAP FIXED BIN INITIAL (1),  /* CHAPTER NUMBER        */
            PARA FIXED BIN INITIAL (1),  /* PARAGRAPH NUMBER      */
            SENT FIXED BIN INITIAL (0),  /* SENTENCE NUMBER       */
            WORD FIXED BIN INITIAL (1),  /* WORD NUMBER           */
            AWORD CHAR (58) VARYING;  /* LENGTH OF LONGEST WORD    */
        DCL OTHER CHAR(58)VAR;
                                   /* TO BE PROCESSED.          */
      DECLARE FIELD CHAR (82) INITIAL (' ');
        DCL LEX FILE RECORD ENV(F(7000, 28));
        DCL C28 CHAR(28);
/*****************************************************************/
 /*                                                               */
/*  THE PROGRAM REQUIRES THAT THE TYPE OF DATA TO BE PROCESSED BE  */
/*  SPECIFIED AS THE FIRST FOUR POSITIONS WITHIN THE 'PARM' FIELD  */
/*  OF THE JCL EXEC CARD OR THE FIRST INPUT RECORD OF SYSIN.  IF   */
/*  THE TYPE PARAMETER IS NOT SPECIFIED THE PROGRAM TERMINATES WITH */
/*  A MSG ON SYSPRINT TO THE USER.  IF THE USER WISHES THE DATA BEING*/
/*  PROCESSED TO BE PRINTED, THE TYPE PARAMETER ON JCL OR SYSIN    */
/*  SHOULD BE FOLLOWED BY A COMMA (,) AND THE PARAMETER 'PRINT' IN */
/*  THE FORMAT 'XXXX,PRINT'.  IF A PRINTOUT IS NOT REQUIRED THE    */
```

```
/*  PARAMETER 'NCPRT' SHCULC BE SPECIFIEC USING THE SAME FORMAT.      */
/*  IF NO PRINT CONTROL PARAMETER IS SPECIFIED, THE DEFAULT IS       */
/*  'NCPRT'.                                                         */
/*                                                                   */
/*  TO USE THE FIRST RECORD OF SYSIN TO INTRODUCE PARAMETERS,        */
/*  THE JCL EXEC PARM FIELD MUST CONTAIN THE VALUE 'FRST'.           */
/*    EXAMPLE USING JCL FOR PARM PASSING;                            */
/*       //   EXEC  PL1,PARM.GO='POET,PRINT'                        */
/*                                                                   */
/*  IF A PLAY HAS STAGE DIRECTIONS THEY WILL BE IGNORED UNLESS       */
/*  THE PARAMETER 'STAGE DIRECTION' IS SPECIFIED.  PARM FIELC FORMAT */
/*  IS:                                                              */
/*           'XXXX,XXXXX,STAGE DIRECTION'                            */
/*                                                                   */
/*    EXAMPLE USING FIRST RECORD OF SYSIN FOR PARAMETERS;            */
/*       //    EXEC  PL1,PARM.GO='FRST'                            */
/*    FIRST CARD OF INPUT ON SYSIN WOULD THEN CONTAIN BEGINING IN    */
/*  CCLUMN 1.  POET,PRINT                                            */
/*                                                                   */
/*  THE FOLLOWING PL1 CODE TESTS THE VALUE PASSED THROUGH JCL        */
/*  IN THE 'PARM' FIELD.  IF THE VALUE = 'FRST' THE FIRST RECORD     */
/*  OF THE INPUT FILE ON SYSIN CONTAINS THE PARAMETERS IN POSITIONS  */
/*  1 THROUGH 10 AND THESE PARAMETERS ARE USED BY THE PROGRAM BY     */
/*  CONTINUING THE PROGRAM AT LABEL 'FIRST'.  IF THE VALUE IN THE    */
/*  'PARM' FIELD IS SOMETHING OTHER THAN 'FRST' THAT VALUE IS USED   */
/*  BY SETTING 'UHLAB' = TO THE VALUE OF THE FIRST FOUR POSITIONS    */
/*  AND PRNTCTRL = THE VALUE OF POSITIONS 6-10 AND CONTINUING THE    */
/*  PROGRAM AT LABEL 'SECOND'.                                       */
/*                                                                   */
 /*                                                                  */
/*                                                                   */
 /*                                                                  */
/*******************************************************************/
          OPEN FILE(SYSIN) INPUT, FILE(SYSPRINT) OUTPUT;
          ON ENDFILE(SYSIN) GO TO OUTOFCRDS;
      IF PARM = 'FRST' THEN GO TO FIRST;
      ELSE  TEMP = PARM;
            UHLAB = SUBSTR(TEMP,1,4);
            PRNTCTRL = SUBSTR(TEMP,6,5);
            IF UHLAB = 'PLAY' | UHLAB = 'FAML' THEN
              STAGEDIRECT = SUBSTR(TEMP,12,15);
            GO TO SECOND;
FIRST:  GET FILE(SYSIN) EDIT (UHLAB,PRNTCTRL,STAGEDIRECT) (A(4),X(1),
          A(5),X(1),A(15));  GET SKIP;
 SECOND:  IF PRNTCTRL = '     ' THEN PRNTCTRL = 'NOPRT'; /*DEFAULT*/
          IF UHLAB = '    ' THEN DO;  /*  DATA TYPE TO BE PROCESSED  */
                                      /* MUST BE SPECIFIED          */
            PUT FILE(SYSPRINT) EDIT ('TYPE OF DATA TO BE PROCESSED ',
                'HAS NOT BEEN SPECIFIED.  PROGRAM TERMINATED.')
                (PAGE,A(29),A(46));
            GO TO CLOSING;
            END;
      PUT FILE(SYSPRINT) EDIT ('TYPE OF DATA TO BE PROCESSED IS ',
                UHLAB) (PAGE,A(32),A(4));
      PUT FILE(SYSPRINT) EDIT (PRNTCTRL,' HAS BEEN SPECIFIED AS ',
                'PRINT CONTROL.','    ') (SKIP,A(5),A(23),A(14),
                SKIP(2),A(4));
      IF UHLAB = 'PLAY' & STAGEDIRECT = 'STAGE DIRECTION' THEN
            PUT FILE(SYSPRINT) EDIT ('PROCESSING OF STAGE DIRECTIONS',
              ' HAS BEEN SPECIFIED.')  (SKIP,A,A);
/*  UHLAB EQUAL TO 'PROS', 'POET', 'PLAY', OR 'SPOK' WILL USE 'COL'  */
/*  INITIAL VALUE OF 73 AS SHOWN ON THE DECLARE STATEMENT FOR 'COL'. */
      IF UHLAB = 'PROS' THEN DO;
```

```
                SENT = 1;
                TOSE = 1;
                GO TO PWA;     /*  PRINT SECTION HEADINGS                  */
                END;
        IF UHLAB = 'PLAY' | UHLAB = 'SPOK' THEN GO TO PWA; /* PRINT
                                        INITIAL SECTION HEADINGS.          */
        IF UHLAB = 'CPLA' THEN COLIM = 68;
   L1: CALL FORM;     /*  SUBROUTINE TO DISTINGUISH WORDS                  */
/*  RESET COUNTERS BASED ON KEYWORD WITHIN TEXT.                           */
        IF UHLAB = 'PROS' THEN DO;
                IF AWORD = '$VOLUME'       THEN DO; L = L7; GO TO L6; END;
                IF AWORD = '$CHAPTER'      THEN DO; L = L8; GO TO L6; END;
                IF AWORD = '$PARAGRAPH'    THEN DO; L = L9; GO TO L6; END;
                IF AWORD = '$SENTENCE'   THEN DO; L = L10; GO TO L6; END;
                GO TO L2;
                END;
        IF UHLAB = 'PLAY' | UHLAB = 'OPLA' THEN DO;
                IF SUBSTR(AWORD,1,1) = '*' &
                   STAGEDIRECT ¬= 'STAGE DIRECTION' THEN GO TO L1;
                IF AWORD = '$ACT'          THEN DO; L = L8; GO TO L6; END;
                IF AWORD = '$SCENE'        THEN DO; L = L9; GO TO L6; END;
                GO TO L2;
                END;
        IF UHLAB = 'SPOK' THEN DO ;
                IF AWORD = '$SERIES'     THEN DO; L = L7; GO TO L6; END;
                IF AWORD = '$SESSION'    THEN DO; L = L8; GO TO L6; END;
                IF AWORD = '$SPEAKER'    THEN DO; L = L9; GO TO L6; END;
/*              GO TO L2;       DEFAULT                                    */
                END;
/*                                                                         */
/*  THE FOLLOWING TRANSFORMS PARAGRAPH, CHAPTER, AND VOLUME END            */
/*  INDICATIONS TO A SINGLE PERIOD '.'.                                    */
/*                                                                         */
 L2:            OTHER = '                    ';   /* CLEAR AREA */
        IF AWORD = '..' | AWORD = '$$$' | AWORD = '$$$$' THEN GO TO P;
 /*
      IF AWORD = '..' | AWORD = '$$$' | AWORD = '$$$$' THEN DO;
        OTHER='.';
                NUMC = 1;
                END;
 */
        OTHER = AWORD;
                TOWC = TOWC + 1;
/*  OUTPUTTING RECORD;  RECORD FORMAT IS:                                  */
/*                                                                         */
/*      POSITIONS 01-04 VARIABLE LENGTH COUNT FIELD.                       */
/*      POSITIONS 05-06 LENGTH OF VARIABLE TEXT WORD.                      */
/*      POSITIONS 07-08 VOLUME NUMBER.                                     */
/*      POSITIONS 09-11 CHAPTER NUMBER.                                    */
/*      POSITIONS 12-14 PARAGRAPH NUMBER.                                  */
/*      POSITIONS 15-19 SENTENCE NUMBER.                                   */
/*      POSITIONS 20-22 WORD NUMBER WITHIN SENTENCE.                       */
/*      POSITIONS 23-80 TEXT WORD, VARYING IN LENGTH UP TO 58 CHAR.        */
/*      PUT STRING(FIELD) EDIT (CNST,'    ',NUMC,VOLU,CHAP,PARA,SENT,
          WORD,OTHER) (F(2),A(4),F(2),F(2),F(3),F(3),F(5),F(3),
          A(NUMC));                                                        */
/* CALL PUTOUT (FIELD)                                                     */
        PUT STRING(FIELD)EDIT(OTHER, CHAP, PARA)(A(20), 2F(4));
        C28 = FIELD;
        WRITE FILE(LEX) FROM(C28);
        PUT SKIP LIST('FIELD=', FIELD);
```

```
/*            DCB=(RECFM=VB,LRECL=80,BLKSIZE=3520)                        */
/*            DDNAME TO BE USED = OUTPUT                                  */
 /*  AT THIS POINT, UHLAB EQUAL TO 'PROS' WILL                            */
 /*  CAUSE THE VARIOUS COUNTERS TO BE INCREMENTED UNDER THE FOLLOWING*/
 /* CONDITIONS AND/OR THE PROGRAM TO BRANCH TO THE LABEL 'PW'.  ALL   */
 /*  OTHER UHLAB VALUES RESULT IN A BRANCH TO PW.                         */
 P:
        IF UHLAB = 'PROS' THEN DO;
              IF AWORD = '.' THEN GO TO CS;      /* SENTENCE            */
              IF AWORD = '?' THEN GO TO CS;      /*  SENTENCE           */
              IF AWORD = ' ' THEN GO TO CS;      /* SENTENCE            */
              END;
        IF UHLAB = 'PROS' | UHLAB = 'PLAY' | UHLAB = 'SPOK'
          THEN DO;
              IF AWORD = '..' THEN GO TO CP;     /*  PARAGRAPH          */
              IF AWORD = '$$$'  THEN GO TO CC;   /* CHAPTER             */
              IF AWORD = '$$$$' THEN GO TO CV;   /*  VOLUME             */
              END;
        GO TO PW;
        CV:   TOVC = TOVC+1;
              VOLU = VOLU+1;
              CHAP =0;
        CC:   TOCH = TOCH+1;
              CHAP = CHAP+1;
              PARA = 0;
        CP:   TOPA = TOPA+1;
              PARA = PARA+1;
              SENT = 0;
        CS:   TOSE = TOSE+1;
              SENT = SENT+1;
              WORD = 0;
        PW:   WORD = WORD+1;
              IF PRNTCTRL = 'NOPRT' THEN GO TO L1;
          IF UHLAB = 'PROS' | UHLAB = 'PLAY' | UHLAB = 'SPOK'
            THEN DO;
              IF AWORD = '..' | AWORD = '$$$' | AWORD = '$$$$'
                 THEN DO;
        PWA:  IF PRNTCTRL = 'NOPRT' THEN GO TO L1;
        PUT FILE(SYSPRINT) EDIT ('VOLU = ','SERIES = ',VOLU,
              '  CHAP = ','ACT = ','ACT = ','  SESSION = ',CHAP,
              '  PARA = ','  SCENE = ','  SCENE = ','  SPEAKER = ',
          PARA,' ') (SKIP(2),A((UHLAB='PROS')*7),
                             A((UHLAB='SPOK')*9),F(2),
/* FORMATS SECTION      */    A((UHLAB='PROS')*9),
/* HEADING BASED ON TYPE*/    A((UHLAB='PLAY')*6),
/* OF PROCESSING        */    A((UHLAB='HAML')*6),
/* SPECIFIED IN FIELD   */    A((UHLAB='SPOK')*12),F(3),
/* 'UHLAB'.             */    A((UHLAB='PROS')*9),
                             A((UHLAB='PLAY')*10),
                             A((UHLAB='HAML')*10),
                             A((UHLAB='SPOK')*12),F(3),
                             A(1));
              GO TO L1;  END;
              END;
          GO TO L1;
        L6:   PUT FILE(SYSPRINT) EDIT ((CARDIMAGE(I) DO I = 1 TO 72))
                 (SKIP(2),72 A(1));  PUT SKIP;
              CALL FORM;
              COL = 73;     /* FORCE READING OF NEW RECORD               */
              GO TO L;
        L7:   VOLU = AWORD;
```

```
        CHAP = 1;  PARA = 1;  SENT = 1;  WORD = 1;
        GO TO PWA;
L8:     CHAP = AWORD;
        PARA = 1;  SENT = 1;  WORD = 1;
        GO TO PWA;
L9:     PARA = AWORD;
        SENT = 1;  WORD = 1;
        GO TO PWA;
L10:    SENT = AWORD;
        WORD = 1;
        GO TO PWA;
OUTOFCRDS: PUT FILE(SYSPRINT) EDIT ('NUMBER OF WORDS ',
        '(CONTENT AND FUNCTION) = ',TOWO)     (PAGE,A(16),A(25),
        F(6));
   IF UHLAB = 'PROS' THEN DO;
   PUT FILE(SYSPRINT) EDIT ('NUMBER OF SENTENCES = ',TOSE)
              (SKIP,A(22),F(6));
   PUT FILE(SYSPRINT) EDIT ('NUMBER OF PARAGRAPHS =',TOPA)
        (SKIP,A(22),F(6));
   PUT FILE(SYSPRINT) EDIT ('NUMBER OF CHAPTERS =     ',TOCH)
        (SKIP,A(24),F(4));
   PUT FILE(SYSPRINT) EDIT ('NUMBER OF VOLUMES =      ',TOVO)
        (SKIP,A(26),F(2));
   GO TO CLOSING;
   END;
   IF UHLAB = 'PLAY' | UHLAB = 'HAML' THEN DO;
   PUT FILE(SYSPRINT) EDIT ('NUMBER OF SENTENCES = ',TOSE)
              (SKIP,A(22),F(6));
   PUT FILE(SYSPRINT) EDIT ('NUMBER OF SCENES = ',TOPA)
              (SKIP,A(19),F(6));
   PUT FILE(SYSPRINT) EDIT ('NUMBER OF ACTS = ',TOCH)
              (SKIP,A(17),F(4));
   GO TO CLOSING;
   END;
   IF UHLAB = 'SPOK' THEN DO;
   PUT FILE(SYSPRINT) EDIT ('NUMBER OF SENTENCES = ',TOSE)
              (SKIP,A(22),F(6));
   PUT FILE(SYSPRINT) EDIT ('NUMBER OF SPEAKERS = ',TOPA)
              (SKIP,A(21),F(6));
   PUT FILE(SYSPRINT) EDIT ('NUMBER OF SESSIONS = ',TOCH)
              (SKIP,A(21),F(4));
   PUT FILE(SYSPRINT) EDIT ('NUMBER OF SERIES = ',TOVO)
              (SKIP,A(19),F(2));
   GO TO CLOSING;
   END;
       PUT FILE(SYSPRINT) EDIT ('NUMBER OF INPUT RECORDS = ',
         TOSE) (SKIP,A(25),F(6));
CLOSING:  CLOSE FILE(SYSIN),                FILE(SYSPRINT);
/* CALL CLSEOUT CLOSES INDX FILE.   */
   GO TO ENDINDX;
/*********************************************************************/
 /*                                                                   */
 /*   SUBROUTINE 'FORM'                                               */
 /*                                                                   */
   FORM: PROCEDURE;
   DCL DUMMY(90) CHAR(1);
         NUMC = 0;
         AWORD = '';
   BUMP: COL = COL+1;
         IF COL < COLIM THEN GO TO EXTRACT;
         IF COL = COLIM THEN DO;
```

```
                  IF CARDIMAGE(COL) = ' ' THEN GO TO LSTCHAR;
                  IF CARDIMAGE(COL) = '-' THEN DO;  /* WORD CONTINUED */
                     WRDCONT = 1;
                     GO TO BUMP;
                     END;
                  IF CARDIMAGE(CCL) = '@' THEN DO;  /* SENTENCE CONT  */
                     LINECONT = 1;
                     GO TO BUMP;
                     END;
                  IF CARDIMAGE(COL) = '#' THEN DO;  /* WORD AND       */
                     WRDCONT = 1;                   /* SENTENCE CONT  */
                     LINECONT = 1;
                     GO TO BUMP;
                     END;
                  END;
               COL = 1;
               IF UHLAB = 'OPLA' THEN GO TO READ68;
      READ:    GET FILE(SYSIN) EDIT ((CARDIMAGE(I) DO I = 1 TO 72),
                     LINEA,SEQUENCE) (72 A(1),A(2),A(6));
         IFLAG =0;
         I=0;
         DO K = 1 TO 71;
/* HYPHENATED WORDS REMAIN HYPHENATED   */
         IF INDEX(',.;?"=() :',CARDIMAGE(K))¬=0 THEN DO;
         IF IFLAG = 1 THEN DO;
         IFLAG =0;
         GO TO E15;
         END;
E5:
         I=I+1;
         DUMMY(I) = ' ';
E10:
         I=I+1;
         DUMMY(I) = CARDIMAGE(K);
         IF CARDIMAGE(K+1) = ' ' THEN GO TO E20;
         IF(CARDIMAGE(K)='.') & ( CARDIMAGE(K+1) = '.') THEN DO;
         IFLAG = 1;
         GO TO E20;
         END;
         I = I+1;
         DUMMY(I) = ' ';
         GO TO E20;
         END;
E15:
         I = I +1;
         DUMMY(I) = CARDIMAGE(K);
E20:
         END;
         I = I +1;
         IF CARDIMAGE(72) = '-' THEN DO;
         DUMMY(I) = '-';
         GO TO E30;
         END;
         DUMMY(I) = ' ';
E30:
         COLIM = I;
         DO J=1 TO COLIM;
         CARDIMAGE(J) = DUMMY(J);
         END;
/*************************************************************************/
            IF UHLAB = 'POET' | UHLAB = 'MILT' THEN DO;
```

```
                IF WRDCONT = 0 THEN WORD = 1;
                 WRDCONT = 0;
                IF LINECONT = 0 THEN DO;
                   SENT = SENT + 1;
                   TOSE = TOSE + 1;
                   END;
                 LINECONT = 0;
                IF UHLAB = 'MILT' THEN VOLU = LINEA;
                IF PRNTCTRL = 'NOPRT' THEN GO TO EXTRACT;
                PUT FILE(SYSPRINT) EDIT ((CARDIMAGE(I) DO I = 1 TO 72),
                   VOLU,'-',SENT) (SKIP,COLUMN(10),72 A(1),X(2),
                   F((UHLAB='MILT')*2),A(UHLAB='MILT'),F(5));
                GO TO EXTRACT;
              END;
              IF UHLAB = 'SPOK' THEN SENT = SENT + 1;
EXTRACT:
                IF CARDIMAGE(COL) = ' ' THEN GO TO LSTCHAR;
                NUMC = NUMC+1;
                AWORD = AWORD || CARDIMAGE(COL);  /*CONCATENATION    */
                GO TO BUMP;
READ68:
                GET FILE(SYSIN) EDIT ((CARDIMAGE(I) DO I= 1 TO 68),
                        LINE,SEQUENCE) (68 A(1),F(4),A(8));
                IF SENT ¬= LINE THEN DO;
                    SENT = LINE;
                    TOSE = TOSE + 1;
                    WORD = 1;
                    END;
                IF PRNTCTRL = 'NOPRT' THEN GO TO EXTRACT;
                PUT FILE(SYSPRINT) EDIT ((CARDIMAGE(I) DO I = 1 TO 68),
                        SENT,SEQUENCE) (SKIP,COLUMN(10),68 A(1),X(2),
                        F(5),X(2),A(8));
                GO TO EXTRACT;
 LSTCHAR:
                IF NUMC = 0 THEN GO TO BUMP;
                RETURN;
                END;
 /*    END OF SUBROUTINE 'FORM'                                          */
 /*********************************************************************/
  ENDINDX:
         END INDX;
//GO.LEX DD UNIT=240O,VOL=SER=ILR1O,
// DCB=(LRECL=28,RECFM=FB,BLKSIZE=7000),DISP=OLD,LABEL=(1,BLP,,OUT)
//GO.SYSIN DD *
PROS,PRINT.
APPENED TO YOU WHEN YOU WERE LOCKED UP WITH THE GHOST."
  "I HAVE NEVER TOLD ANY ONE, CECIL," SAID VIRGINIA GRAVELY.
  "I KNOW THAT, BUT YOU MIGHT TELL ME."
  "PLEASE DON"T ASK ME, CECIL, I CANNOT TELL YOU. POOR SIR SIMON  I OWE
HIM A GREAT DEAL. YES, DON"T LAUGH, CECIL, I REALLY DO. HE MADE ME SEE
WHAT LIFE IS, AND WHAT DEATH SIGNIFIES, AND WHY LOVE IS STRONGER THAN B-
OTH." ..
  THE DUKE ROSE AND KISSED HIS WIFE LOVINGLY.
  "YOU CAN HAVE YOUR SECRET AS LONG AS I HAVE YOUR HEART," HE MURMURED.
  "YOU HAVE ALWAYS HAD THAT, CECIL."
  "AND YOU WILL TELL OUR CHILDREN SOME DAY, WON"T YOU?" ..
  VIRGINIA BLUSHED.
/*
//
```

2.8 Sample Output

2.8.1 Word Index

		3	6,	3	6,	3	6,	3	7,	3	7,	3	7,	3	7,	3	7,	3	8,	4	1,
		4	1,	4	1,	4	2,	4	2,	4	2,	4	2,	4	2,	4	2,	4	2,	4	3,
		4	3,	4	3,	4	3,	4	3,	4	4,	4	4,	4	5,	4	6,	4	6,	4	6,
		4	6,	4	6,	5	1,	5	1,	5	3,	5	3,	5	3,	5	5,	5	6,	5	7,
		5	7,	5	7,	5	7,	6	1,	6	1,	6	1,	6	1,	6	2,	6	2,	6	2,
		6	2,	6	2,	6	2,	6	2,	6	2,	6	2,	6	3,	6	3,	6	3,	6	3,
		6	3,	6	3,	7	1,	7	1,	7	1,	7	1,	7	2,	7	2,	7	2,	7	2,
		7	2,	7	4,	7	5,	7	5,												
WITHERED	3	3	7,	5	2,	6	3,														
WITHOUT	2	6	2,	7	4,																
WOMAN	4	1	2,	1	4,	1	6,	1	6,												
WON	5	1	6,	6	1,	6	1,	6	2,	7	6,										
WONDER	2	5	1,	6	3,																
WONDERFUL	3	1	2,	1	2,	4	5,														
WONDERFULLY	1	7	1,																		
WOOD	1	1	3,																		
WOODS	2	4	1,	5	1,																
WORD	1	7	1,																		
WORDS	5	1	6,	2	2,	3	7,	5	7,	7	4,										
WORE	1	4	1,																		
WORK	3	4	5,	5	1,	7	2,														
WORLD	3	1	1,	4	2,	4	2,														
WORLD-SOUL	1	2	2,																		
WORM-EATEN	1	4	1,																		
WORN	1	3	4,																		
WORTH	1	7	2,																		
WORTHY	3	1	2,	4	5,	7	3,														
WOULD	23	1	1,	1	6,	1	6,	3	1,	3	2,	3	4,	3	5,	3	7,	3	8,	4	2,
		4	3,	4	5,	4	6,	5	1,	5	1,	6	1,	6	1,	6	2,	6	2,	7	2,
		7	2,	7	2,	7	3,														
WOUNDED	1	4	5,																		
WRAPPER	1	2	2,																		
WRAPS	1	1	4,																		
WRETCHED	1	2	4,																		
WRINKLED	1	3	6,																		
WRISTS	2	2	2,	3	5,																
WRITHED	1	3	6,																		
WRITING	1	3	6,																		
WRITING-MASTER	1	3	3,																		
WRONG	1	5	1,																		
WROTE	1	4	5,																		
WROUGHT	1	3	7,																		
WRUNG	1	5	6,																		
YARD	1	6	2,																		
YE	3	3	7,	3	7,	3	7,														
YEAR	4	4	2,	4	6,	4	6,	6	1,												
YEARS	7	1	6,	2	4,	4	2,	4	5,	5	1,	5	1,	7	1,						
YELLOW	2	4	3,	5	1,																
YES	4	1	5,	5	3,	7	5,	7	5,												
YESTERDAY	1	5	3,																		
YET	2	3	4,	7	2,																
YEW-TREE	3	3	6,	5	2,	7	1,														
YORK	3	1	2,	2	2,	5	1,														
YOU	86	1	1,	1	1,	1	1,	1	1,	1	1,	1	4,	2	2,	2	2,	2	2,	2	2,
		3	3,	3	3,	3	3,	5	1,	5	1,	5	1,	5	1,	5	1,	5	1,	5	1,
		5	1,	5	1,	5	1,	5	1,	5	1,	5	1,	5	1,	5	1,	5	1,	5	1,
		5	1,	5	1,	5	1,	5	1,	5	1,	5	1,	5	1,	5	1,	5	1,	5	1,
		5	2,	5	2,	5	3,	5	3,	5	3,	5	3,	5	5,	5	5,	5	5,	5	5,
		5	5,	5	6,	5	7,	6	1,	6	1,	6	1,	6	2,	6	2,	6	2,	6	2,
		6	2,	6	2,	6	2,	6	2,	6	3,	7	2,	7	2,	7	2,	7	2,	7	2,

2.8.2 Word Frequency List

1101	,
732	THE
446	.
435	AND
329	OF
322	"
280	A
271	TO
196	HE
189	WAS
186	IN
151	HIS
122	HAD
114	WITH
113	THAT
112	IT
112	ON
102	AS
100	AT
100	I
98	FOR
98	HER
86	YOU
63	THEY
61	OTIS
57	HAVE
56	BY
54	SHE
52	BEEN
49	NOT
49	WERE
48	BUT
47	;
47	ALL
47	BE
47	VIRGINIA
46	HIM
46	IS
45	FROM
45	UP
44	GHOST
44	NO
43	S
41	CANTERVILLE
39	THERE
39	VERY
38	ONE
36	LITTLE
36	OUT
36	WHO
35	MR
35	SO
35	WHICH
34	WHEN
32	SAID
31	
30	OLD
30	QUITE
30	THEM
29	MY
28	FAMILY
28	THEIR
27	AFTER

3. Program CONCORD

3.1 Identification

Title	CONCORD
User Group	Institute of Library Research
Category	CIS Text Processing
Source Language	PL/1
Machine Configuration	IBM/360 Model 91, printer one magnetic tape and one disk.
Space Requirements	Step Region
	PL1L 162K
	GO 150K
	Core used 140K
Author	G. Silva
Date	March 1970

3.2 Purpose

To read input text from cards in stream form.

To produce a concordance in KWIC format[1], with the option to drop words on an exclusion list.

3.3 Output record format

The output record format of the indexing phase is fixed length, 148 character records. Each record contains a sort key plus a segment made up of the word surrounded by context.

---

[1] A concordance is an alphabetically ordered list of the occurrences of either all, or a selection of items in a text, each occurrence surrounded by an appropriate portion of its context, together with a numerical reference to each passage where the item occurs.

The sortkey occupies the first 24 bytes of the record and is followed by the word in context. Details of the output format are given below:

Positions 1 - 20 - Word

Positions 21 - 24 - Four-byte chapter number

Positions 25 - 28 - Four-byte paragraph number

Positions 29 - 148 - Word within context

Output of the records is accomplished by using the PL/1 PUT STRING function as for the Indexing program.

These records are input to the sort phase where they are arranged into ascending order by the sortkey.

The printed output record (concordance) is illustrated in Section IV. 3.8.

3.4 Checkout

The concordance was tested on Oscar Wilde's "The Canterville Ghost", a short story of a total of 11761 words (punctuation marks are not counted).

3.4.1 Timing

The concordance was made in two steps: 7989 words were indexed and output to magnetic tape (first step) to serve as input to the dictionary sort (second step).

Indexing phase took	28.77s cpu time
Sort took	27.14s cpu time
Total cpu time	55.91s

i.e., 4.7s/100 input words

3.4.2 Error Checking

None

3.4.3 Cost

The cost for the whole operation was:

Indexing phase	$11.49
Dictionary sort	$18.20
Total	$29.69

i.e., approximately $2.60 /1000 words

3.4.4 Remarks

In its present form the program accepts input text in a much simpler format than described in section IV.1. In fact, it can only recognize chapters and paragraphs. Additional coding is needed to allow it to cope with the full range of Program Control key words and indexing information allowed for in program INDX.

Program CONCORD has an option to exclude words selected by the user, and supplied to the program in the form of data. This option to exclude words was used in the final test on "The Canterville Ghost". Out of the 11761 words of text, 3772 were excluded, mainly articles, conjunctions, pronouns, forms of the verbs to be and to have, and some modals.

3.5 Program Description

Main Steps

The program reads in a parameter which indicates whether the exclusion options is required or not. If the option is required, the program reads in the exclusion list from cards and sorts it into alphabetical order using subroutine READIN and SORT.

Then, the text itself is read in card by card and stored in a character string PARA, which will contain one paragraph of text at a time. Everytime a new card is read the program checks (1) for a hyphen in column 72, and (2) for two consecutive periods (..), the signal for the end of a paragraph.

When the end of a paragraph is reached, the program begins to process the string PARA. The string is divided into individual words, and, if the exclusion switch is on, each word is tested against the exclusion list. This is done by means of subroutine BIS, which uses a binary search technique. If it is not found on the exclusion list, the word, its indexing information, and the word surrounded by some context are put out to tape in a form suitable for sorting.

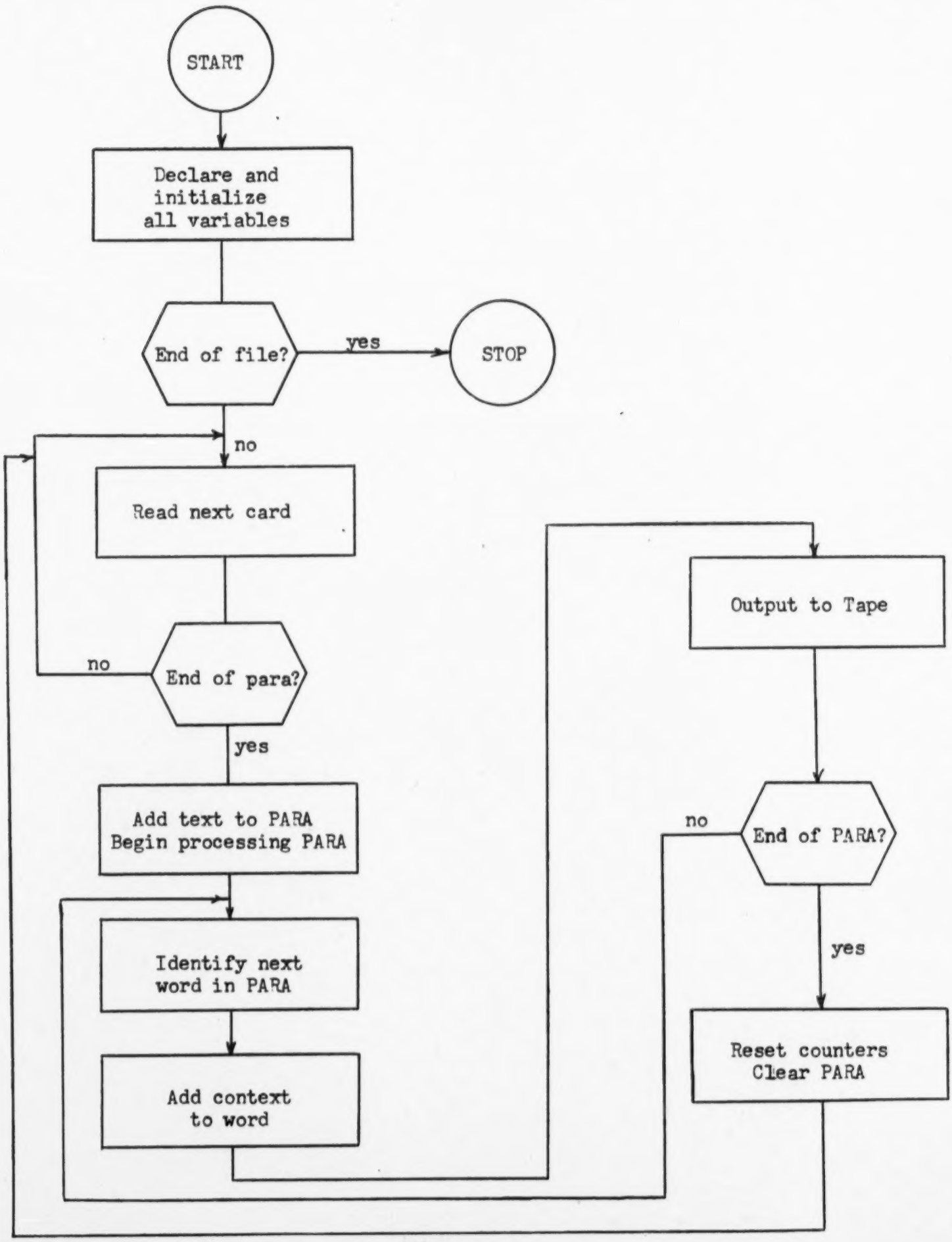
When the string PARA is fully processed, the program returns to construct the next paragraph from text.

When the entire input text has been processed in this fashion, the resulting tape file is input to program ALPHA which produces the concordance.

Program ALPHA is described in detail in subsection IV. 4.1.

3.6 Flow Chart For Program CONCORD

Indexing Phase



3.7 Program Listing

```
//NRB409GR  JOB  '5M,10KI,R',SILVA
// EXEC PL1,PG='(50,20)'
//PL1L.SYSLIN  DD  SPACE=(TRK,(20,10))
//PL1L.SYSUT1  DD  SPACE=(TRK,(20,10))
//PL1L.SYSUT3  DD  SPACE=(TRK,(20,10))
//PL1L.SYSIN DD *
 CONCORD: PROC OPTIONS(MAIN);
          DCL (DUMMY) CHAR(80)VAR;
          DCL PARA CHAR(10000)VAR;
          DCL (TOCH, TCPA) BIN FIXED;
          DCL TOWO BIN FIXED;
          DCL (FRONT, IFRONT) CHAR(50)VAR;
          DCL BACK CHAR(70);
          DCL (FRSTLTR, LSTLTR) BIN FIXED;
          DCL LEN FIXED BIN;
          DCL WORD CHAR(20)VAR;
          DCL TEXT CHAR(120);
          DCL ULINE CHAR(80)VAR INIT((80)'_');
          DCL EXCL_LIST(300) CHAR(82)VAR;
          DCL C20 CHAR(20);
          DCL C148 CHAR(148);
          DCL LEX FILE RECORD ENV(F(6956,148));
          ON ENDFILE(SYSIN) GO TO FINIS;
          GET LIST(IEXCLUDE);      GET SKIP;
          IF IEXCLUDE=0 THEN GO TO START;
          CALL READIN(EXCL_LIST, NO_EXCL);
          GET SKIP;
          CALL SORT(NO_EXCL, EXCL_LIST);
          PUT SKIP LIST('EXCLUSION LIST');
          DO J=1 TO NO_EXCL;
          PUT SKIP LIST(EXCL_LIST(J));
          END;
  START:
          TOWO=0;
          TOPA = 0;
          TOCH = 1;
NEXTPA:
          TOPA = TOPA+1;
          PARA = '';
NEXTLIN:
          GET EDIT(DUMMY)(A(80));
          LEN = 72;
/*  TEST FOR HYPHEN IN COL 72 */
          IF SUBSTR(DUMMY, LEN, 1) = '-' THEN DO;
          PARA = PARA || SUBSTR(DUMMY, 1, 71);
          GO TO NEXTLIN;
          END;
LOOP1:
          LEN = LEN-1;
          IF SUBSTR(DUMMY, LEN, 1) = ' ' THEN GO TO LOOP1;
/* LEN = POSITION OF LAST NON-BLANK CHARACTER ON CARD */
          IF (SUBSTR(DUMMY, LEN, 1)='.') & (SUBSTR(DUMMY, LEN-1, 1)='.')
          THEN DO;
          PARA = PARA || SUBSTR(DUMMY, 1, LEN-2);
CONTEXT:
/* PRINTS WORDS OF TEXT IN CONTEXT */
          LEN = LENGTH(PARA);
STEP1:
/*  FIND NEXT WORD AND WORDLENGTH  */
          K = 1;
SEARCH:
```

```
        FRONT = (50) ' ';
        BACK = (70) ' ';
        IFRONT = (50) ' ';
        DO J=1 BY 1 WHILE(K<=LEN);
        CALL NI(WORD, FRSTLTR, LSTLTR, PARA, LEN);
        IF FRSTLTR=LEN THEN GO TO NEXTPA;
        IF WORD = '$$$' THEN DO;
        TOCH = TOCH+1;
        TOPA=1;
        GO TO STEP5;
        END;
        IF IEXCLUDE=0 THEN GO TO ADD;
        CALL BIS(EXCL_LIST, NO_EXCL, WORD, JFOUND);
        IF JFOUND = 1 THEN GO TO STEP5;
 ADD:
        LENG = LSTLTR - FRSTLTR;
STEP2:
/*  CONTEXT BEFORE WORD  */
        IF FRSTLTR<51 THEN DO;
        IFRONT = SUBSTR(PARA, 1, FRSTLTR-1);
        IBLNK = 51-FRSTLTR;
        FRONT = SUBSTR(FRONT, 1, IBLNK) || IFRONT;
        GO TO STEP3;
        END;
        FRONT = SUBSTR(PARA, FRSTLTR-50, 50);
STEP3:
/*  CONTEXT AFTER WORD  */
        IF LEN-FRSTLTR < 70 THEN DO;
        BACK = SUBSTR(PARA, FRSTLTR);
        GO TO STEP4;
        END;
        BACK = SUBSTR(PARA, FRSTLTR, 70);
STEP4:
/*  PRINT WORD IN CONTEXT  */
        TEXT = FRONT || BACK;
        TOWO = TOWO +1;
        C20=WORD;
        PUT STRING(C148)EDIT(C20,TOCH,TOPA,TEXT)(A(20),2 F(4),A(120));
        WRITE FILE(LEX) FROM(C148);
        PUT SKIP LIST(TEXT);
        PUT EDIT(ULINE)(SKIP(0),COL(51),A(LENG));
 STEP5:
        K = LSTLTR+1;
        GO TO SEARCH;
        END;
        GO TO NEXTPA;
        END;
        PARA = PARA || SUBSTR(DUMMY, 1, LEN) || ' ';
        GO TO NEXTLIN;
 BIS:
        PROC( LIST, LENGTH, ITEM, JFOUND);
 /*  BIS STANDS FOR " BINARY SEARCH "  */
        /*  BIS RETURNS JFOUND=1 IF ELEMENT FOUND, ZERO, IF NOT  */
        DCL ITEM CHAR(*)VAR;
        DCL LIST(*       ) CHARACTER(* )VAR;
        LOWER=1;
        IUPPER = LENGTH + 1;
 BIO:
        MID = (LOWER + IUPPER)/2;
        IF ITEM = LIST(MID) THEN DO;
        JFOUND=1;
```

```
        RETURN;
        END;
        IF ITEM > LIST(MID) THEN DO;
             IF(MID = LOWER)|(MID = IUPPER ) THEN DO;
             JFOUND=0;
             RETURN;
             END;
        LOWER = MID;
        GO TO B1C;
                                    END;
             IF (MID = LOWER)| ( MID = IUPPER) THEN DO;
             JFOUND=C;
             RETURN;
             END;
        IUPPER=MID;
        GO TO B10;
 END BIS;
NI:
        PROC(TEMP, FRSTLTR, LSTLTR, DUMMY, LEN);
        DCL TEMP CHAR(*)VAR,
        (FRSTLTR, LSTLTR, LEN)BIN FIXED,
        DUMMY CHAR(*)VAR;
/*  LOOPS WHEN PUNCTUATION ENCOUNTERED  */
        DO FRSTLTR = K TO LEN WHILE
        (INDEX(' ,.;?"=()-:',SUBSTR(DUMMY, FRSTLTR, 1)) ¬=0);
        IF FRSTLTR = LEN THEN RETURN;
        END;
/*  LOOPS WHEN NO PUNCTUATION ENCOUNTERED, TO FIND WRDLGTH  */
        DO LSTLTR = FRSTLTR+1 TO LEN WHILE
        (INDEX(' ,.;?"=()-', SUBSTR(DUMMY, LSTLTR, 1)) = 0);
        END;
        TEMP = SUBSTR(DUMMY, FRSTLTR, (LSTLTR-FRSTLTR));
        RETURN;
        END NI;
READIN:
        PROC(LINE,NUM);
/*  READS IN A LIST OF TERMS TYPED ONE PER CARD  */
/* DATA DECK MUST END WITH A ZERO IN COL 1  */
/* NO OTHER FORMAT RESTRICTIONS ARE IMPOSED  */
        DCL LINE (*)CHAR(*)VAR;
        DCL I FIXED BIN;
        PUT SKIP LIST('INPUT LIST');
        N=0;
R10:
        N=N+1;
        GET EDIT(LINE(N)) (A(80));
        IF LINE(N) = 'C' THEN GO TO R50;
/* NORMALIZE CARD FORMAT  */
/* ADD BLANK AT THE BEGINNING AND AT THE END OF  EACH LINE  */
        LINE(N) = ' '||LINE(N)||' ';
/* REDUCE MULTIPLE BLANKS TO ONE BLANK  */
R20:
        I = INDEX(LINE(N),'  ');
        IF I¬=0 THEN DO ;
             IF I=LENGTH(LINE(N))-1  THEN DO ;
             LINE(N) = SUBSTR(LINE(N),1,I);
             GO TO R30;
             END;
        LINE(N) = SUBSTR(LINE(N),1,I)||SUBSTR(LINE(N),I+2);
        GO TO R20;
        END;
```

```
 R30:
/*  REMOVE LEADING AND TRAILING BLANK  */
        LINE(N)=SUBSTR(LINE(N), 2, LENGTH(LINE(N))-2);
        PUT SKIP LIST(LINE(N));
        GO TO R10;
 R50:
        NUM = N-1;
        RETURN;
 END READIN;
 SORT:
        PROC(N,ITEM);
        DCL ITEM(* )CHAR(* )VAR;
        DCL TEMP CHAR(80)VAR;
        NUM = N-1;
        J=0;
 S1:    J=J+1;
        IF J>NUM THEN GO TO S10;
        IF ITEM(J) > ITEM(J+1) THEN DO;
 S2:
        TEMP= ITEM(J);
        ITEM(J) = ITEM(J+1);
        ITEM(J+1) = TEMP;
        ISWAP=1;
        GO TO S1;
                        END;
        GO TO S1;
 S10:
        IF ISWAP =0 THEN GO TO S20;
        ISWAP=0;
        NUM = NUM -1;
        J=0;
        GO TO S1;
  S20:
        PUT SKIP LIST('NO OF ITEMS SORTED=',N);
        RETURN;
  END SORT;
 FINIS:
        PUT SKIP LIST('TOTAL NO OF WORDS PROCESSED= ' , TOWO);
        END CONCORD;
/*
//GO.LEX DD UNIT=2400,VOL=SER=ILRC90,
// DCB=(LRECL=148,RECFM=FB,BLKSIZE=6956),DISP=OLD,LABEL=(1,BLP,,OUT)
//GO.SYSIN DD *
1
HAVE
HAS
HAD
DO
DID
CAN
COULD
SHALL
SHOULD
WILL
WOULD
MAY
MIGHT
MUST
MR
MRS
SIR
```

```
WAS
WERE
MY
YOUR
HIS
WON
YES
NC
YE
N
CHASE
HERS
OUR
THEIR
ME
HER
HIM
US
THEM
THE
A
AN
B
R
DE
S
T
O
AND
OR
BLT
AM
ARE
IS
I
YOU
HE
SHE
IT
WE
THEY
C
```

THE CANTERVILLE GHOST

WHEN MR. HIRAM B. OTIS, THE AMERICAN MINISTER, BOUGHT CANTERVILLE CHASE , EVERY ONE TOLD HIM HE WAS DOING A VERY FOOLISH THING, AS THERE WAS NO DOUBT AT ALL THAT THE PLACE WAS HAUNTED. INDEED, LORD CANTERVILLE HIMSELF, WHO WAS A MAN OF THE MOST PUNCTILIOUS HONOUR, HAD FELT IT HIS DUTY TO MENTION THE FACT TO MR. OTIS, WHEN THEY CAME TO DISCUSS TERMS.

"WE HAVE NOT CARED TO LIVE IN THE PLACE OURSELVES," SAID LORD CANTERVILLE, "SINCE MY GRAND-AUNT, THE DOWAGER DUCHESS OF BOLTON, WAS FRIGHTENED INTO A FIT, FROM WHICH SHE NEVER REALLY RECOVERED, BY TWO SKELETON HANDS BEING PLACED ON HER SHOULDERS AS SHE WAS DRESSING FOR DINNER, AND I FEEL BOUND TO TELL YOU, MR. OTIS, THAT THE GHOST HAS BEEN SEEN BY SEV-

```
/*
//
```

```
THE OFFICE, HOWEVER, WAS LOCKED, FOR THE CHIEF CONSTABLE HAD GONE HOME. THE ONLY PERSON LEFT IN THE BUILDING WAS A POLICEMAN WHO WAS ON DUTY AT THE DOOR, AND HE WAS STILL IN THE HOUSE, AND THOUGH NOW ALMOST AN INVALID, WAS BY
BUSY THAT HE WOULD PLAY DICE WITH THE CARETAKER. THE CARETAKER, WHO WAS ALSO ONE OF THE POLICE AGENTS WHO HAD CHARGE OF THE CAPTURED
LECTIONS OF THE POLICE AGENT WHO HAD NOT LOST M
YOUR WIFE." "WELL, I OUGHT ADMIT IT." SAID THE
IMMENSE SUPPLIES, HOWEVER, IT WAS THE CANTEEN-
I DID KILL HER." "STAYE YOU TO PLAYING ON," HE
CALLY. "NO RUINS. AN OBSTACLES " ANSWERED THE
EMERALD-GREEN BLOOD?" "WELL, REALLY," SAID THE
DEATHER, AND MUCH MORE THAN THAT TO HAVE A FAVILY
OKED UP INTO HIS OLD WITHERED FACE. "PLOP, DEAR
                                    THEM THE
VIRGINIA MADE NO ANSWER, AND THE
LITTLE VIRGINIA," THEY CRIED, "GET BACK " PUT THE
NIA, REWARE WE MAY NEVER SEE YOU AGAIN," PUT THE
G PULLING AT HER DRESS. "QUICK, QUICK," CRIED THE
ALLY FELT THAT, AS SHE HAD BEEN FRIGHTENED BY THE
E A NIGHTINGALE BECAN TO SING. SHE THOUGHT OF THE
TERVIEW WITH HIM ON THE SUBJECT OF THE "JEWELS THE
ET, MR. OTIS, THAT YOU TOOK THE EMPLIRE AND THE
AT A VALUATION, AND ANYTHING THAT DEPENDED TO THE
P TO ALLOW HIS DAUGHTER TO DEIAIN THE PRESENT THE
HAPPENED TO YOU WHEN YOU WERE LOCKED UP WITH THE
THESE PRACTICAL JOKES AND MORE." "EXCEPT ON THE
ANY MORE." "EXCEPT ON THE GHOST  EXCEPT ON THE
FARFUL WORDS:                    YE OTIS
UPED THE HONEST SOUL THAT THEY WERE NOT AFRAID OF
EN TOO DOGMATIC IN HIS DENIAL OF THE EXISTENCE OF
HT SUPPER. THE CLEVERESTATION IN NE WAY TURNED UPON
AT HIS HEAD  IT WAS QUITE UNPARABLE. RESTLESS, NO
THE GRISLY PHANTOM, FEELING THAT, AFTER ALL, TWO
WAS TYPE THE WEW-TREE SPREADS OLT ITS
MOON LOOKS DOWN, AND THE YEW-TREE SPREADS OLT ITS
O MAKE HIS WAY QUIETLY TO WASHINGTON OTIS'S ROOM,
UTY TO APPEAR IN THE CREPICRE ONCE A WEEK, AND TO
UBEN, OR THE STRANGLED BABE," HIS DEBUT AS MOUNT
MELY SENSIBLE, MISS VIRGINIA E. OTIS WAS A LITTLE
OKING ROUND IN ASTONISHMENT AT THE PRETTY LITTLE
ONLY SIX LINES:                  WHEN A GOLDEN
LIBRARY WINDOW?"  "OH," DEREN," CRIED THE LITTLE
THE COUNTY," TELLING THEM TO LOOK OUT FOR A LITTLE
SPENDING SEVERAL WINTERS IN POSTON WHEN SHE WAS A
, WHICH IS THE REWARD OF ALL GOOD LITTLE AMERICAN
DE THERE OF LADY CANTERVILLE'S FRENCH CHAPERESSE TO
STAIRCASE HE RECOVERED HIMSELF, AND DERERMINED TO
AST SEVEN, THE APRIVAL OF THE HOUSEMAIDS MADE HIM
HE KEPT HIS ROOM, AND AT LAST MADE UP HIS MIND TO
T SUCCESS. I KNOW LOTS OF PEOPLE THERE WHO WOULD
VE YOUR MIND. WY FATHER WILL BE ONLY TOO HAPPY TO
NLY REMEMBERED THAT, SOME FEW DAYS BEFORE, HE HAD
TH HIM ON THE SUBJECT OF THE JEWELS THE GHOST HAD
HIS CAUGHTER TO RETAIN THE PRESENT THE GHOST HAD
FOR ONE OF HER SEVEN INVARIED CAUGHTERS. ANY HAP
MONY REARS,                AND A LITTLE CHILD
HTER, SHE IS MERELY A CHILD, AND HAS AS YET, I AM
AS TREARLY FORGHTENED, AND, AFTER A SECOND MASTY
WILL FIND IT A MOST EXCELLENT DINNER." THE GHOST
LACK OAK, AT THE END OF WHICH WAS A LARGE STAINED-
HAD FRIGHTENED INTO A FIT AS SHE STOOD BEFORE THE
WITH ITS RIGHT HAND, IT BORE ALOFT A CALDRIDY OF
IGHT HAD ENTIRELY EXPED FROM ITS HOLLOW EYES, THE
IFE WERE BLAZONED IN AZURE AND OCHRE ON HIS ON HE
```

4. Sorting Procedures

4.1 Program Description

Procedure GAMMA requires a sorting technique which can handle large amounts of data and be readily modifiable to accommodate various data formats. Both of these requirements are met by the IBM sort/merge program, which GAMMA invokes through the subprogram IHESRT. IHESRT is called at different entry points according to the source and destination of the sort records; the calling parameters describe the records and define the fields on which sorting is to be performed.

In the first sort, records are retrieved from a data set and returned to GAMMA. This is accomplished by calling IHESRTC, which passes records composed of a 20-character word and an 8-character identification from a SORTIN data set to sort/merge to be sorted into ascending sequence. The sorted records are returned one at a time to entry point ENT35 in GAMMA, where once the frequency of each word is determined, the word, its frequency, and associated identifications are printed. Additionally, a 28-character record containing the word and its frequency is written to a temporary data set in a form suitable for printing.

When all of the records returned by IHESRTC have been processed, GAMMA calls IHESRTA. This routine retrieves the 28-character records created at ENT35, calls IBM sort/merge to sort the records alphabetically within each frequency and into descending sequence by frequency, and prints the sorted records.

Paralleling the first sort in GAMMA, procedure ALPHA sorts 148-character records into ascending sequency by a 20-character

word and an 8-character identification. It invokes sort/merge through IHESRTC, which returns the sorted records to ENT35. The text and identification are printed and the keyword of the text is underlined.

Experience derived from testing GAMMA indicates that when IHESRT must interface with the calling program, for instance where IHESRTC returns sorted records at ENT35, machine time is very high. If acquiring a large region is more economical than writing to and reading from disk, it is desirable to give sort/merge at least two or three times the minimum main storage required. Further, the amount of buffer space needed by sort/merge to block records efficiently increases with the number of intermediate disk storage areas supplied.

Reference

IBM System/360 Operating System Sort/Merge, Form C28-6543-5